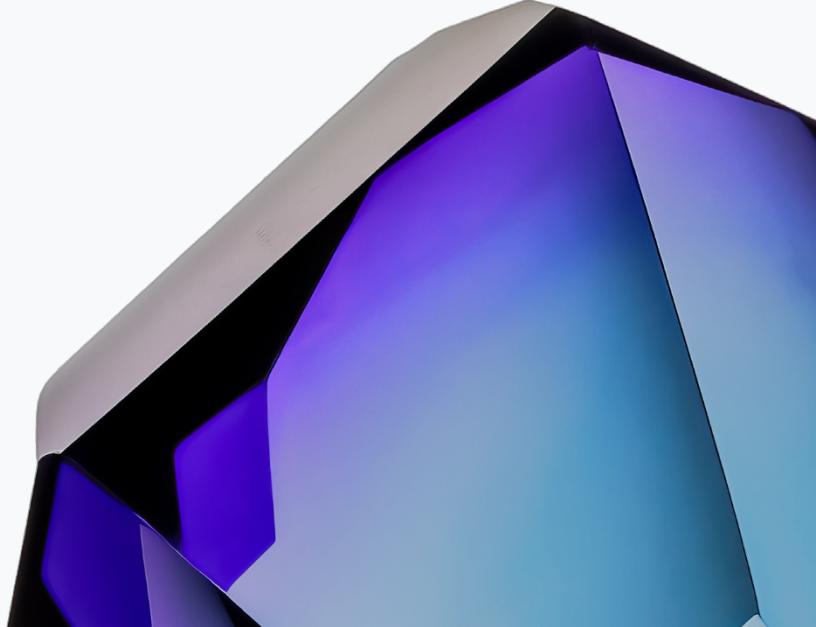


Введение в агенты

Авторы: Алан Блаунт, Антонио Гулли, Шубхам Сабу, Майкл Циммерман и Владимир Вускович



Благодарности

Авторы

Энрике Чан Майк

Кларк Дерек Иган

Анант Навалгария

Канчана Патлолла

Джулия Визингер

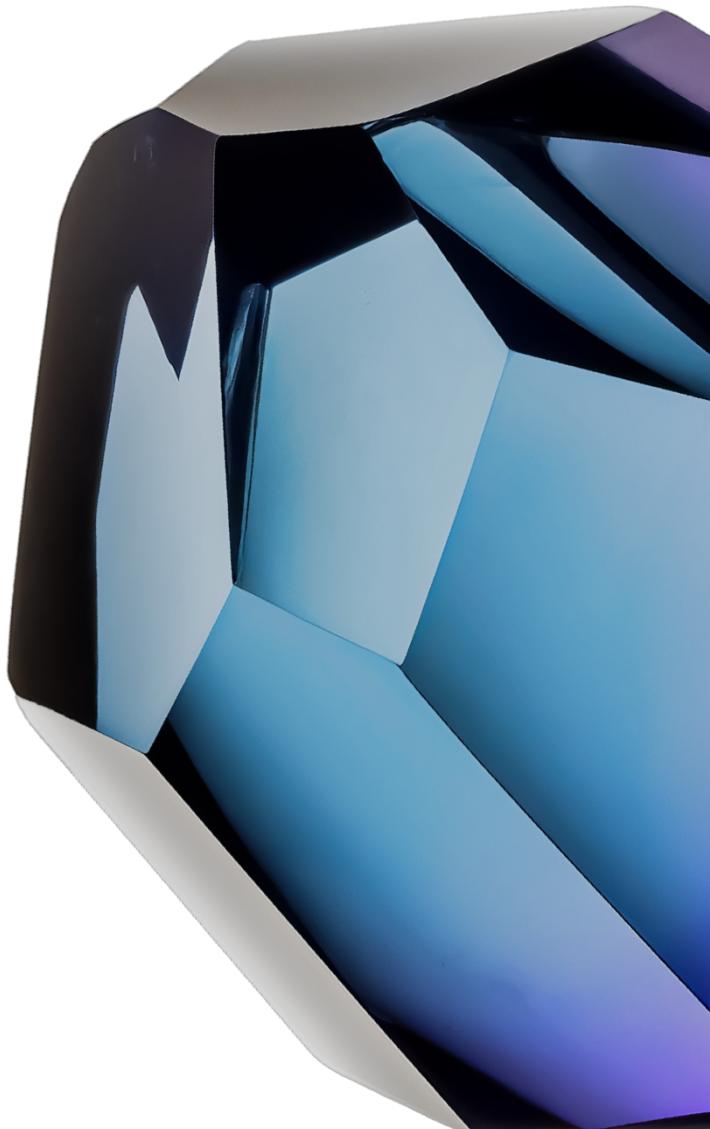
Кураторы и редакторы

Анант Навалгария Канчана

Патлолла

Дизайнер

Майкл Лэннинг



Содержание

От прогнозирующего ИИ к автономным агентам	6
Введение в ИИ-агенты	8
Процесс решения проблем с помощью агентов	10
Таксономия агентных систем	14
Уровень 0: Основная система рассуждений	15
Уровень 1: Связанный решатель проблем	15
Уровень 2: Стратегическое решение проблем	16
Уровень 3: Коллаборативная мультиагентная система	17
Уровень 4: Саморазвивающаяся система	18
Архитектура основного агента: модель, инструменты и оркестрация	19
Модель: «мозг» вашего ИИ-агента	19
Инструменты: «руки» вашего ИИ-агента	20
Поиск информации: опора на реальность	21
Выполнение действий: изменение мира	21
Вызов функций: подключение инструментов к вашему агенту	22
Агенты и деньги	22
Уровень оркестрации	22
Основные проектные решения	23
Обучайте с использованием знаний в области и персонажей	23
Работайте с контекстом	24
Используйте шаблоны и шаблонные системы	24
Разработка агентов и услуги	26
Агенты для экспериментации: структурированный подход к непредсказуемому	27
Измените то, что имеет значение: инструментируйте успех как A/B-эксперимент	29
Качества, которые «прошел/не прошел»: использование LM Judge	29
Разработайте на основе метрик: ваше решение о внедрении или отказе от внедрения	30
Отладка с помощью трассировок OpenTelemetry: ответ на вопрос «Почему?»	30
Цените ошибки и ошибки: руководство по автоматизации	31

Содержание

Взаимодействие агентов	31
Агенты и люди	32
Агенты и агенты	33
	34

Содержание

Обеспечение безопасности единственного агента: компромисс между доверием и безопасностью	34
Идентичность агента: новый класс принципала	35
Политики ограничения доступа	37
Защита агента ADK	37
Масштабирование от одного агента до корпоративного парка	39
Безопасность и конфиденциальность: укрепление границ агента	40
Управление агентами: плоскость управления вместо разброса	40
Как агенты развиваются и учатся	42
Как агенты учатся и саморазвиваются	43
Моделирование и Agent Gym — следующий рубеж	46
Примеры продвинутых агентов	47
Со-ученый Google	47
Агент AlphaEvolve	49
Заключение	51
Примечания	52



Агенты являются естественным развитием языковых моделей, которые нашли применение в программном обеспечении.

От прогнозирующего ИИ к автономным агентам

Искусственный интеллект меняется. В течение многих лет основное внимание уделялось моделям, которые преуспевают в пассивных, дискретных задачах: отвечают на вопросы, переводят текст или генерируют изображения по запросу. Эта парадигма, хотя и является мощной, требует постоянного человеческого управления на каждом этапе. Сейчас мы наблюдаем смену парадигмы, переход от ИИ, который просто предсказывает или создает контент, к новому классу программного обеспечения, способного автономно решать проблемы и выполнять задачи.

Эта новая граница построена вокруг агентов ИИ. Агент — это не просто модель ИИ в статическом рабочем процессе, это полноценное приложение, которое составляет планы и предпринимает действия для достижения целей. Оно сочетает в себе способность языковой модели (LM) к **рассуждению** с практической способностью **действовать**, позволяя

ему справляться со сложными многоэтапными задачами, которые модель сама по себе не может выполнить. Важнейшая способность заключается в том, что агенты могут работать самостоятельно, выясняя следующие шаги, необходимые для достижения цели, без постоянного руководства со стороны человека.

Этот документ является первым в серии из пяти частей и служит официальным руководством для разработчиков, архитекторов и руководителей продуктов, переходящих от доказательств концепции к надежным агентным системам производственного уровня. Хотя создание простого прототипа не представляет сложности, обеспечение безопасности, качества и надежности является серьезной задачей. В данной статье представлены исчерпывающие основы:

- **Основная анатомия:** разложение агента на три основных компонента: модель рассуждений, инструменты для действий и управляющий уровень оркестрации.
- **Таксономия возможностей:** классификация агентов от простых, связанных между собой систем решения проблем до сложных, совместных мультиагентных систем.
- **Архитектурный дизайн:** углубленное изучение практических аспектов проектирования каждого компонента, от выбора модели до внедрения инструментов.
- **Создание для производства:** установление дисциплины Agent Ops, необходимой для оценки, отладки, обеспечения безопасности и масштабирования агентских систем от одного экземпляра до целого парка с корпоративным управлением.

Основываясь на предыдущем [техническом документе по агентам](#)¹ и [Agent Companion](#)², это руководство предоставляет основополагающие концепции и стратегические рамки, необходимые для успешного создания, развертывания и управления этим новым поколением интеллектуальных приложений, которые могут рассуждать, действовать и наблюдать для достижения [целей](#)³.

Слов недостаточно, чтобы описать, как люди взаимодействуют с ИИ. Мы склонны антропоморфизировать и использовать человеческие термины, такие как «думать», «рассуждать» и «знать». У нас еще нет слов для обозначения «знать с семантическим значением» и «знать с высокой вероятностью максимизации функции вознаграждения». Это два разных типа знания, но результаты в 99,9% случаев одинаковы.

Введение в ИИ-агенты

В простейших терминах ИИ-агент можно определить как комбинацию моделей, инструментов, уровня оркестрации и служб выполнения, которые используют LM в цикле для достижения цели. Эти четыре элемента составляют основную архитектуру любой автономной системы.

- **Модель («мозг»):** основная языковая модель (LM) или базовая модель, которая служит центральным механизмом рассуждений агента для обработки информации, оценки вариантов и принятия решений. Тип модели (общего назначения, точно настроенная или мультимодальная) определяет когнитивные способности агента. Агентная система является конечным куратором окна контекста ввода LM.
- **Инструменты («руки»):** эти механизмы связывают рассуждения агента с внешним миром, позволяя ему выполнять действия, выходящие за рамки генерации текста. К ним относятся расширения API, функции кода и хранилища данных (такие как базы данных или векторные хранилища) для доступа к фактической информации в режиме реального времени. Агентная система позволяет LM планировать, какие инструменты использовать, запускает инструмент и помещает результаты его работы в окно входного контекста следующего вызова LM.
- **Уровень оркестрации («нервная система»):** управляющий процесс, который управляет операционным циклом агента. Он обрабатывает планирование, память (состояние) и выполнение стратегии рассуждений. Этот уровень использует рамки подсказок и методы рассуждений (такие как

[Chain-of-Thought⁴](#) или [ReAct⁵](#)) для разбиения сложных целей на этапы и принятия решения, когда нужно думать, а когда использовать инструмент. Этот уровень также отвечает за предоставление агентам памяти для «запоминания».

- **Развертывание («тело и ноги»):** хотя создание агента на ноутбуке эффективно для прототипирования, именно развертывание в производственной среде делает его надежным и доступным сервисом. Это включает в себя размещение агента на безопасном, масштабируемом сервере и его интеграцию с основными производственными сервисами для мониторинга, ведения журналов и управления. После развертывания пользователи могут получить доступ к агенту через графический интерфейс, а другие агенты — программно через API Agent-to-Agent (A2A).

В конечном итоге, создание генеративного агента ИИ — это новый способ разработки решений для решения задач. Традиционный разработчик действует как «каменщик», точно определяя каждый логический шаг. Разработчик агента, напротив, больше похож на режиссера. Вместо того, чтобы писать явный код для каждого действия, вы устанавливаете сцену (руководящие инструкции и подсказки), выбираете актеров (инструменты и API) и предоставляете необходимый контекст (данные). Основной задачей становится руководство этим автономным «актером», чтобы он выполнил запланированное действие.

Вы быстро обнаружите, что самая большая сила LM — его невероятная гибкость — также является вашей самой большой головной болью. Способность большой языковой модели делать что угодно затрудняет заставить ее надежно и идеально выполнять одну конкретную задачу. То, что мы раньше называли «инжинирингом подсказок», а теперь называем «инжинирингом контекста», направляет LM на генерацию желаемого результата. Для любого отдельного вызова LM мы вводим наши инструкции, факты, доступные инструменты для вызова, примеры, историю сеансов, профиль пользователя и т. д., заполняя окно контекста именно той информацией, которая необходима для получения нужных результатов. Агенты — это программное обеспечение, которое управляет входными данными LM для выполнения работы.

Отладка становится необходимой, когда возникают проблемы. «Agent Ops» по сути переопределяет привычный цикл измерения, анализа и оптимизации системы. С помощью трассировок и журналов вы можете отслеживать «процесс мышления» агента, чтобы выявить отклонения от намеченного пути выполнения. По мере развития моделей и совершенствования фреймворков роль разработчика заключается в предоставлении

важнейшие компоненты: опыт в данной области, четко определенную индивидуальность и беспроблемную интеграцию с инструментами, необходимыми для выполнения практических задач. Важно помнить, что комплексные оценки и анализы часто перевешивают влияние первоначального запроса.

Когда агент точно настроен с четкими инструкциями, надежными инструментами и интегрированным контекстом, служащим памятью, отличным пользовательским интерфейсом, способностью планировать и решать проблемы, а также общими знаниями о мире, он выходит за рамки понятия простой «автоматизации рабочего процесса». Он начинает функционировать как совместная сущность: высокоеффективный, уникально адаптируемый и чрезвычайно способный новый член вашей команды.

По сути, агент — это система, предназначенная для курирования контекстного окна. Это представляет собой непрерывный цикл сборки контекста, запроса модели, наблюдения за результатом и повторной сборки контекста для следующего шага. Контекст может включать системные инструкции, ввод пользователя, историю сеанса, долговременную память, базовые знания из авторитетных источников, информацию о том, какие инструменты можно использовать, и результаты уже запущенных инструментов. Такое сложное управление вниманием модели позволяет ее способностям к рассуждению решать проблемы в новых обстоятельствах и достигать целей.

Процесс решения проблем с помощью агентов

Мы определили ИИ-агент как полное, ориентированное на цель приложение, которое объединяет модель рассуждений, инструменты для действий и управляющий уровень оркестрации. Версия Shofi — это «LM в цикле с инструментами для достижения цели».

Но как на самом деле *работает* эта система? Что делает агент с момента получения запроса до момента предоставления результата?

В своей основе агент работает по непрерывному циклическому процессу для достижения своих целей. Хотя этот цикл может быть очень сложным, его можно разбить на пять основных этапов, как подробно описано в книге «*Agentic System Design*»⁽⁶⁾:

1. **Получение задания:** процесс запускается конкретной целью высокого уровня. Это задание предоставляется пользователем (например, «Организовать поездку моей команды на предстоящую конференцию») или автоматизированным триггером (например, «Поступил новый билет высокой приоритетности от клиента»).
2. **Сканирование ситуации:** агент воспринимает свое окружение, чтобы собрать контекст. Это включает в себя доступ уровня оркестрации к доступным ресурсам: «Что говорится в запросе пользователя?», «Какая информация находится в моей краткосрочной памяти? Я уже пытался выполнить эту задачу? Пользователь давал мне указания на прошлой неделе?», «К чему я могу получить доступ с помощью своих инструментов, таких как календари, базы данных или API?».
3. **Обдумайте:** это основной цикл «мышления» агента, основанный на модели рассуждений. Агент анализирует **задачу** (шаг 1) в контексте **ситуации** (шаг 2) и разрабатывает план. Это не единичная мысль, а часто цепочка рассуждений: «Чтобы забронировать поездку, мне сначала нужно узнать, кто входит в команду. Я воспользуюсь инструментом `get_team_roster`. Затем мне нужно будет проверить их доступность через `calendar_api`».
4. **Принятие мер:** уровень оркестрации выполняет первый конкретный шаг плана. Он выбирает и вызывает соответствующий инструмент — вызов API, запуск функции кода или запрос к базе данных. Это агент, *действующий* в мире за пределами своего внутреннего мышления.
5. **Наблюдать и повторять:** агент наблюдает за *результатом* своих действий. Инструмент `get_team_roster` возвращает список из пяти имен. Эта новая информация добавляется в контекст или «память» агента. Затем цикл повторяется, возвращаясь к шагу 3: «Теперь, когда у меня есть список, мой следующий шаг — проверить календарь для этих пяти человек. Я буду использовать `calendar_api`».

Этот цикл «Думай, действуй, наблюдай» продолжается — он управляет **уровнем оркестрации**, обосновывается **моделью** и выполняется **инструментами** до тех пор, пока внутренний план агента не будет выполнен и первоначальная **миссия** не будет достигнута.

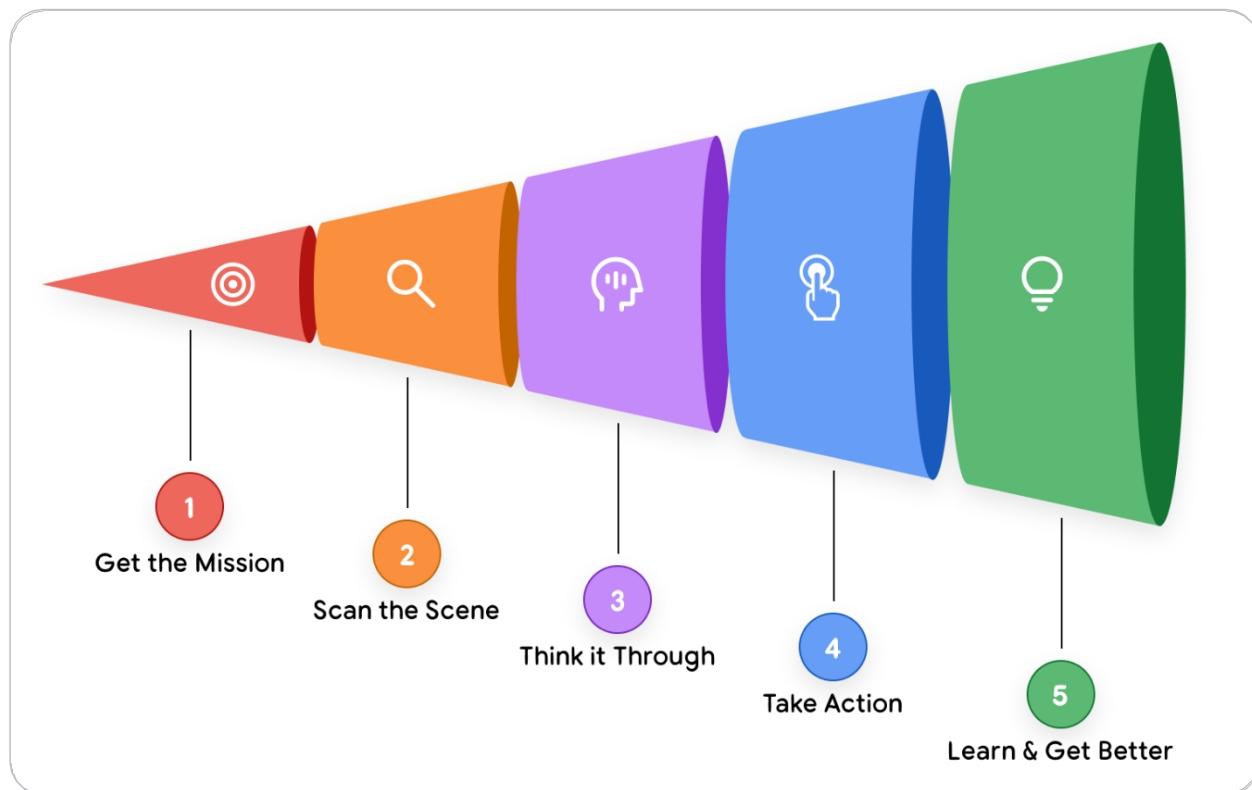


Рисунок 1: Процесс решения проблем с помощью агентного ИИ

Давайте рассмотрим реальный пример того, как агент службы поддержки клиентов будет работать в этом 5-этапном цикле:

Представьте, что пользователь спрашивает: «Где мой заказ № 12345?»

Вместо того чтобы сразу же приступить к действиям, агент сначала переходит в фазу «**Обдумывания**», чтобы разработать полную стратегию. Он рассуждает:

«Пользователь хочет узнать статус доставки. Чтобы дать полный ответ, мне нужен план из нескольких шагов:

1. *Идентификация:* сначала я должен найти заказ в нашей внутренней базе данных, чтобы подтвердить его существование и получить подробную информацию.
2. *Отслеживание:* из этих сведений мне нужно извлечь номер отслеживания перевозчика. Затем я запрошу API внешнего перевозчика для получения текущего статуса.
3. *Отчет:* наконец, я должен обобщить собранную информацию в четкий и полезный ответ для пользователя».

Имея в виду этот **многоэтапный план**, агент приступает к его выполнению.

На первом этапе «Act» он выполняет первый шаг своего плана, вызывая инструмент `find_order("12345")`. Он наблюдает за результатом — полной записью заказа, включая номер отслеживания «ZYX987».

Уровень оркестрации агента распознает, что первый шаг его плана выполнен, и немедленно переходит ко второму. Он действует, вызывая инструмент `get_shipping_status("ZYX987")`. Он наблюдает новый результат: «Out for Delivery» (Отправлено на доставку).

Наконец, успешно выполнив этапы сбора данных своего плана, агент переходит к шагу «Report». Он понимает, что у него есть все необходимые компоненты, планирует окончательное сообщение и действует, генерируя ответ: «Ваш заказ № 12345 находится в процессе доставки!

Таксономия агентских систем

Понимание 5-ступенчатого операционного цикла — это первая часть головоломки. Вторая — признание того, что этот цикл можно масштабировать по сложности для создания различных классов агентов. Для архитектора или руководителя продукта ключевым первоначальным решением является определение *того*, какой тип агента следует создать.

Мы можем классифицировать агентские системы по нескольким общим уровням, каждый из которых основывается на возможностях предыдущего.



Рисунок 2: Агентная система в 5 шагах

Уровень 0: Основная система рассуждений

Прежде чем мы сможем создать агента, мы должны оснастить его «мозгом» в его наиболее базовой форме: самим механизмом рассуждения. В этой конфигурации языковая модель (LM) работает изолированно, реагируя исключительно на основе своих обширных предварительно обученных знаний, без каких-либо инструментов, памяти или взаимодействия с реальной средой.

Ее сила заключается в обширной подготовке, которая позволяет ей объяснять устоявшиеся концепции и планировать подход к решению проблемы с большой глубиной. Компромиссом является полное отсутствие осознания реальности в режиме реального времени; она функционально «слепа» к любым событиям или фактам, выходящим за рамки ее обучающих данных.

Например, она может объяснить правила профессионального бейсбола и полную историю команды «Нью-Йорк Янкис». Но если вы спросите: «Какой был окончательный счет вчерашней игры «Янкис»?», она не сможет ответить. Эта игра — конкретное событие реального мира, которое произошло после сбора данных для обучения, поэтому эта информация просто отсутствует в ее знаниях.

Уровень 1: Связанный решатель проблем

На этом уровне механизм рассуждений становится функциональным агентом, подключаясь к внешним инструментам и используя их — это компонент «Руки» нашей архитектуры. Его решение проблем больше не ограничивается статическими, заранее обученными знаниями.

Используя 5-ступенчатый цикл, агент теперь может ответить на наш предыдущий вопрос. Учитывая «задачу»: «Каков был окончательный счет вчерашней игры Yankees?», его этап «Размышление» распознает это как потребность в данных в режиме реального времени. Затем его этап «Действие» вызывает инструмент, такой как Google Search API, с соответствующей датой и поисковыми терминами. Он «наблюдает» за результатом поиска (например, «Янки выиграли 5:3») и синтезирует этот факт в окончательный ответ.

Эта фундаментальная способность взаимодействовать с миром — будь то использование поискового инструмента для получения результатов, финансового API для получения текущих цен на акции или базы данных через Retrieval-Augmented Generation (RAG) — является основной способностью агента уровня 1.

Уровень 2: Стратегическое решение проблем

Уровень 2 знаменует собой значительное расширение возможностей, переход от выполнения простых задач к стратегическому планированию сложных, многосторонних целей. Ключевым навыком, который появляется на этом уровне, является **контекстная инженерия**: способность агента активно выбирать, объединять и управлять наиболее релевантной информацией для каждого этапа своего плана.

Точность агента зависит от сфокусированного, высококачественного контекста. Контекстная инженерия курирует ограниченное внимание модели, чтобы предотвратить перегрузку и обеспечить эффективную работу.

Рассмотрим, например, «миссию»: **«Найти хорошую кофейню на полпути между моим офисом по адресу 1600 Amphitheatre Parkway, Маунтин-Вью, и офисом моего клиента по адресу 1 Market St, Сан-Франциско».**

Агент уровня 2 начнет создавать план:

1. **Подумает**: «Сначала я должен найти середину пути».
 - **Действие**: вызов инструмента «Карты» с обоими адресами.
 - **Наблюдать**: «Середина пути — это Миллбрей, Калифорния».
2. **Подумать**: «Теперь я должен найти кофейни в Миллбрей. Пользователь просил найти «хорошие» кофейни, поэтому я буду искать заведения с рейтингом 4 звезды и выше».

- **Действие:** вызовите инструмент `google_places` с запросом `query="кофейня в Миллбрей, Калифорния", min_rating=4.0`. (Это контекстная инженерия — она автоматически создала новый целенаправленный поисковый запрос на основе результатов предыдущего шага).
- **Наблюдайте:** «Поиск возвращает результаты «Millbrae Coffee» и «The Daily Grind».

3. Подумайте: «Я обобщу эти результаты и представлю их пользователю».

Такое стратегическое планирование также позволяет оказывать проактивную помощь, например, агент, который читает длинное электронное письмо с подтверждением рейса, анализирует ключевой контекст (номер рейса, дата) и добавляет его в ваш календарь.

Уровень 3: Коллaborативная мультиагентная система

На самом высоком уровне парадигма полностью меняется. Мы отходим от создания единого всемогущего «суперагента» и переходим к «команде специалистов», работающих в `concefi`, модели, которая напрямую отражает человеческую организацию. Коллективная сила системы заключается в этом разделении труда.

Здесь агенты рассматривают других агентов как инструменты. Представьте себе агента «Менеджер проекта», получающего «Задание»: «Запустить в продажу наши новые наушники «Solaris».

Агент-менеджер проекта не выполняет всю работу самостоятельно. Он действует, создавая новые миссии для своей команды специализированных агентов, подобно тому, как это происходит в реальной жизни:

1. **Делегаты MarketResearchAgent:** «Проанализируйте цены конкурентов на наушники с функцией шумоподавления. К завтрашнему дню предоставьте сводный документ».
2. **Делегирует MarketingAgent:** «Подготовьте три варианта пресс-релиза, используя спецификацию продукта «Solaris» в качестве контекста».

3. **Делегирует WebDevAgent:** «Создайте HTML-код новой страницы продукта на основе приложенных макетов дизайна».

Эта модель совместной работы, хотя и ограничена в настоящее время ограничениями современных LM, представляет собой передовую технологию автоматизации всех сложных бизнес-процессов от начала до конца.

Уровень 4: Саморазвивающаяся система

Уровень 4 представляет собой глубокий скачок от делегирования к автономному созданию и адаптации. На этом уровне агентская система может выявлять пробелы в своих собственных возможностях и динамически создавать новые инструменты или даже новых агентов для их заполнения. Она переходит от использования фиксированного набора ресурсов к их активному расширению.

Следуя нашему примеру, агент «Проектный менеджер», которому поручено запуск «Solaris», может понять, что ему необходимо отслеживать настроения в социальных сетях, но в его команде нет такого инструмента или агента.
его команде.

1. **Думать (мета-рассуждение):** «Я должен отслеживать обсуждение «Solaris» в социальных сетях, но у меня нет такой возможности».
2. **Действие (автономное создание):** вместо того, чтобы потерпеть неудачу, он вызывает высокоуровневый инструмент AgentCreator с новой миссией: «Создать нового агента, который будет отслеживать социальные сети на наличие ключевых слов «наушники Solaris», проводить анализ настроений и предоставлять ежедневный отчет».
3. **Наблюдайте:** новый специализированный SentimentAnalysisAgent создается, тестируется и добавляется в команду на лету, готовый внести свой вклад в выполнение первоначальной задачи.

Такой уровень автономности, при котором система может динамически расширять свои собственные возможности, превращает команду агентов в по-настоящему обучающуюся и развивающуюся организацию.

Архитектура основного агента: модель, инструменты и оркестрация

Мы знаем, что делает агент и как он может масштабироваться. Но как его на самом деле построить? Переход от концепции к коду заключается в конкретной архитектурной конструкции его трех основных компонентов.

Модель: «мозг» вашего ИИ-агента

LM — это ядро мышления вашего агента, и его выбор является критически важным архитектурным решением, которое определяет когнитивные способности, эксплуатационные затраты и скорость вашего агента. Однако рассматривать этот выбор как простой вопрос выбора модели с самым высоким результатом тестирования — это обычный путь к провалу. Успех агента в производственной среде редко определяется общими академическими тестами.

Реальный успех требует модели, которая превосходит другие по **основным характеристикам агента**: превосходные **способности к рассуждению** для решения сложных многоэтапных задач и надежное **использование инструментов** для взаимодействия с окружающим [миром](#)⁷.

Чтобы сделать это хорошо, определите бизнес-проблему, а затем протестируйте модели по метрикам, которые напрямую соотносятся с этим результатом. Если вашему агенту нужно писать код, протестируйте его на вашей частной кодовой базе. Если он обрабатывает страховые заявления, оцените его способность извлекать информацию из ваших конкретных форматов документов. Затем этот анализ необходимо сопоставить с практическими аспектами стоимости и задержки. «Лучшая» модель — это та, которая находится на оптимальном пересечении качества, скорости и цены для [вашей конкретной задачи](#)⁸.

Вы можете выбрать более одной модели, «команду специалистов». Не стоит использовать кувалду, чтобы расколоть орех. Надежная архитектура агента может использовать передовую модель, такую как **Gemini 2.5 Pro**, для выполнения сложных задач первоначального планирования и сложных рассуждений, но затем интеллектуально направлять более простые задачи с большим объемом данных, такие как классификация намерений пользователей или резюмирование текста, в гораздо более быструю и экономичную модель, такую как **Gemini 2.5 Flash**. Маршрутизация моделей может быть автоматической или жестко запрограммированной, но она является ключевой стратегией для оптимизации как [производительности, так и затрат](#)⁽⁹⁾.

Тот же принцип применяется к обработке различных типов данных. В то время как нативная мультимодальная модель, такая как [Gemini live mode](#)⁽¹⁰⁾, предлагает оптимизированный путь для обработки изображений и аудио, альтернативой является использование специализированных инструментов, таких как [Cloud Vision API](#)⁽¹¹⁾ или [Speech-to-Text API](#)⁽¹²⁾. В этом случае мир сначала преобразуется в текст, который затем передается в модель, основанную только на языке, для логического вывода. Это добавляет гибкости и позволяет использовать лучшие в своем классе компоненты, но также значительно усложняет процесс.

Наконец, сфера искусственного интеллекта находится в состоянии постоянной и быстрой эволюции. Модель, которую вы выберете сегодня, через шесть месяцев будет устаревшей. Подход «настроил и забыл» является неустойчивым. Создание системы с учетом этой реальности означает инвестирование в гибкую операционную структуру — [практику](#) «Agent Ops»⁽¹³⁾. Благодаря надежному конвейеру CI/CD, который постоянно оценивает новые модели по ключевым бизнес-показателям, вы можете снизить риски и ускорить обновления, гарантируя, что ваш агент всегда будет работать на основе лучших доступных интеллектуальных ресурсов, без необходимости полной переработки архитектуры.

Инструменты: «руки» вашего ИИ-агента

Если модель — это мозг агента, то инструменты — это руки, которые связывают его мышление с реальностью. Они позволяют агенту выйти за пределы статических обучающих данных, чтобы получать информацию в режиме реального времени и действовать в мире. Надежный интерфейс инструмента представляет собой цикл из трех этапов: определение того, что может делать инструмент, его вызов и наблюдение за результатом.

Вот несколько основных типов инструментов, которые разработчики предоставят своим агентам. Более подробную информацию можно найти в техническом документе, посвященном инструментам для агентов, из этой серии.

Поиск информации: опора на реальность

Самым основным инструментом является возможность доступа к актуальной информации.

Технология RAG (Retrieval-Augmented Generation) дает агенту «библиотечную карточку» для запроса внешних знаний, часто хранящихся в **векторных базах данных** или **графах знаний**, от внутренних документов компании до веб-знаний через Google Search. Для структурированных данных инструменты **Natural Language to SQL (NL2SQL)** позволяют агенту запрашивать базы данных для ответа на аналитические вопросы, такие как «Какие продукты были самыми продаваемыми в прошлом квартале?». Изучая информацию перед тем, как ответить — будь то в документе или базе данных — агент опирается на факты, что значительно снижая количество ошибок.

Выполнение действий: изменение мира

Истинная сила агентов раскрывается, когда они переходят от чтения информации к активным действиям. Объединяя существующие **API** и функции кода в виде инструментов, агент может отправлять электронные письма, планировать встречи или обновлять записи о клиентах в ServiceNow. Для более динамичных задач агент может **писать и выполнять код на лету**. В безопасной песочнице он может генерировать SQL-запрос или скрипт Python для решения сложной проблемы или выполнения вычислений, превращаясь из знающего помощника в [автономного участника](#)⁽¹⁴⁾.

Сюда также входят инструменты для взаимодействия с людьми. Агент может использовать инструмент **Human in the Loop (HITL)** для приостановки своего рабочего процесса и запроса подтверждения (например, `ask_for_confirmation()`) или запроса конкретной информации из пользовательского интерфейса (например, `ask_for_date_input()`), обеспечивая участие человека в принятии важных решений. HITL может быть реализован с помощью SMS-сообщений и задачи в базе данных.

Вызов функций: подключение инструментов к вашему агенту

Чтобы агент мог надежно выполнять «вызов функций» и использовать инструменты, ему необходимы четкие инструкции, безопасные соединения и [оркестрация](#)¹⁵. Это обеспечивают давно существующие стандарты, такие как спецификация **OpenAPI**, предоставляющая агенту структурированный контракт, в котором описываются назначение инструмента, необходимые параметры и ожидаемый ответ. Эта схема позволяет модели каждый раз генерировать правильный вызов функции и интерпретировать ответ API. Для упрощения обнаружения и подключения к инструментам стали популярны открытые стандарты, такие как **Model Context Protocol (MCP)**, поскольку они более [удобны](#)⁽¹⁶⁾. Кроме того, некоторые модели имеют встроенные инструменты, такие как Gemini с встроенным Google Search, где вызов функции происходит как часть [самого](#) вызова LM⁽¹⁷⁾.

Уровень оркестрации

Если модель — это мозг агента, а инструменты — его руки, то уровень оркестрации — это центральная нервная система, которая их соединяет. Это двигатель, который запускает цикл «Думай, действуй, наблюдать», состояние машины, которая управляет поведением агента, и место, где тщательно разработанная логика разработчика оживает. Этот уровень — не просто трубопровод; это дирижер всей симфонии агента, который решает, когда модель должна рассуждать, какой инструмент должен действовать и как результаты этого действия должны повлиять на следующее движение.

Основные варианты проектирования

Первое архитектурное решение — определение степени автономности агента. Выбор существует в широком диапазоне. С одной стороны, у вас есть детерминированные, предсказуемые рабочие процессы, которые вызывают LM в качестве инструмента для выполнения конкретной задачи — небольшое добавление ИИ для улучшения существующего процесса. С другой стороны, у вас есть LM в роли водителя, который динамически адаптируется, планирует и выполняет задачи для достижения цели.

Параллельный выбор — это метод реализации. Конструкторы без кода обеспечивают скорость и доступность, позволяя бизнес-пользователям автоматизировать структурированные задачи и быстро создавать простых агентов. Для более сложных, критически важных систем, такие фреймворки, как [Agent Development Kit \(ADK\) от Google](#)¹⁸, обеспечивают глубокий контроль, настраиваемость и возможности интеграции, необходимые инженерам.

Независимо от подхода, необходим фреймворк производственного уровня. Он должен быть **открытым**, позволяя подключать любую модель или инструмент, чтобы избежать привязки к поставщику. Он должен обеспечивать точный контроль, позволяя использовать гибридный подход, при котором недетерминированное мышление LM регулируется жестко запрограммированными бизнес-правилами. Наиболее важно, что фреймворк должен быть построен с **учетом наблюдаемости**. Когда агент ведет себя непредсказуемо, вы не можете просто поставить точку останова в «мысли» модели. Надежный фреймворк генерирует подробные трассировки и журналы, раскрывая всю траекторию рассуждений: внутренний монолог модели, выбранный ею инструмент, сгенерированные ею параметры и наблюдаемый ею результат.

Инструктируйте с помощью знаний в области и персонажа

В рамках этой структуры самым мощным рычагом разработчика является обучение агента с использованием знаний в конкретной области и четкой персоны. Это достигается с помощью системного запроса или набора основных инструкций. Это не просто команда, это конституция агента.

Здесь вы говорите ему: «Ты — полезный агент службы поддержки клиентов Acme Corp...» и предоставляете ограничения, желаемую схему вывода, правила взаимодействия, конкретный тон голоса и четкие указания о том, когда и почему он должен использовать свои инструменты. Несколько примеров сценариев в инструкциях обычно очень эффективны.

Дополнение контекстом

«Память» агента организована в контекстном окне LM во время выполнения. Для более полного погружения в тему см. технический документ, посвященный памяти агента, в этой серии.

Краткосрочная память — это активная «записная книжка» агента, в которой хранится история текущего разговора. Она отслеживает последовательность пар (действие, наблюдение) из текущего цикла, предоставляемая моделью необходимый контекст для принятия решения о дальнейших действиях. Это может быть реализовано в виде абстракций, таких как состояние, аффакты, сессии или потоки.

Долговременная память обеспечивает постоянство между сессиями. С архитектурной точки зрения это почти всегда реализуется как еще один специализированный инструмент — система RAG, подключенная к векторной базе данных или поисковой системе. Оркестратор дает агенту возможность предварительно загружать и активно запрашивать свою собственную историю, позволяя ему «запоминать» предпочтения пользователя или результат аналогичной задачи, выполненной несколько недель назад, для обеспечения действительно персонализированного и непрерывного опыта.¹⁹

Мультиагентные системы и принципы проектирования

По мере усложнения задач создание единого всемогущего «суперагента» становится неэффективным. Более эффективным решением является использование подхода «команды специалистов», который отражает человеческую организацию. Это ядро **мультиагентной системы**: сложный процесс

разбиты на отдельные подзадачи, и каждая из них поручается специальному специализированному агенту ИИ. Такое разделение труда позволяет сделать каждого агента более простым, сфокусированным и легким в создании, тестировании и обслуживании, что идеально подходит для динамичных или длительных бизнес-процессов.

Архитекторы могут полагаться на проверенные шаблоны проектирования агентов, хотя возможности агентов и, следовательно, шаблоны [быстро](#) развиваются⁽²⁰⁾. Для динамических или нелинейных задач необходим шаблон «Координатор». Он вводит агента-«менеджера», который анализирует сложный запрос, сегментирует основную задачу и интеллектуально направляет каждую подзадачу соответствующему специализированному агенту (например, исследователю, писателю или программисту). Затем координатор агрегирует ответы каждого специалиста, чтобы сформулировать окончательный, исчерпывающий ответ.

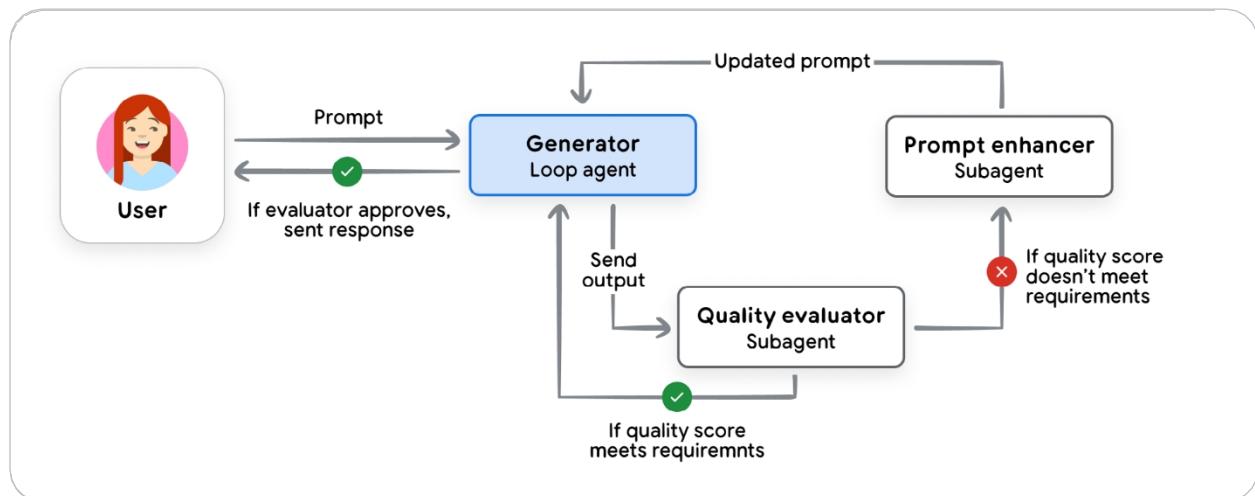


Рисунок 3: Шаблон «итеративное уточнение» из
<https://cloud.google.com/architecture/choose-design-paflern-agentic-ai-system>

Для более линейных рабочих процессов лучше подходит **последовательный** паттерн, действующий как цифровая сборочная линия, где выходные данные одного агента становятся прямыми входными данными для следующего. Другие ключевые паттерны сосредоточены на качестве и безопасности. Паттерн «**Итеративное уточнение**» создает цикл обратной связи, используя агента-«генератора» для создания контента и агента-«критика» для его оценки по

стандартами качества. Для задач с высокими ставками критически важным является паттерн «Человек в цикле» (**HITL**), создающий намеренную паузу в рабочем процессе, чтобы получить одобрение от человека, прежде чем агент предпримет значительные действия.

Развертывание агента и услуги

После создания локального агента вам понадобится развернуть его на сервере, где он будет работать постоянно и где другие люди и агенты смогут им пользоваться. Продолжая нашу аналогию, развертывание и услуги будут телом и ногами нашего агента. Для эффективной работы агенту требуется несколько услуг, история сеансов, сохранение памяти и многое другое. Как создатель агента, вы также будете отвечать за то, что регистрировать в журнале, и какие меры безопасности принимать для обеспечения конфиденциальности данных, хранения данных и соблюдения нормативных требований. Все эти услуги входят в сферу деятельности при развертывании агентов в производственной среде.

К счастью, разработчики агентов могут полагаться на десятилетия опыта в области инфраструктуры хостинга приложений. В конце концов, агенты — это новая форма программного обеспечения, и ко многим из них применимы те же принципы. Разработчики могут полагаться на специально разработанные, специфичные для агентов варианты развертывания, такие как **VefieX AI Agent Engine**, который поддерживает среду выполнения и все остальное в одной [платформе](#)²¹. Для разработчиков программного обеспечения, которые хотят более непосредственно контролировать свои стеки приложений или развертывать агенты в рамках существующей инфраструктуры DevOps, любой агент и большинство служб агентов могут быть добавлены в контейнер Docker и развернуты в стандартных для отрасли средах выполнения, таких как [Cloud Run](#) или [GKE](#)²².

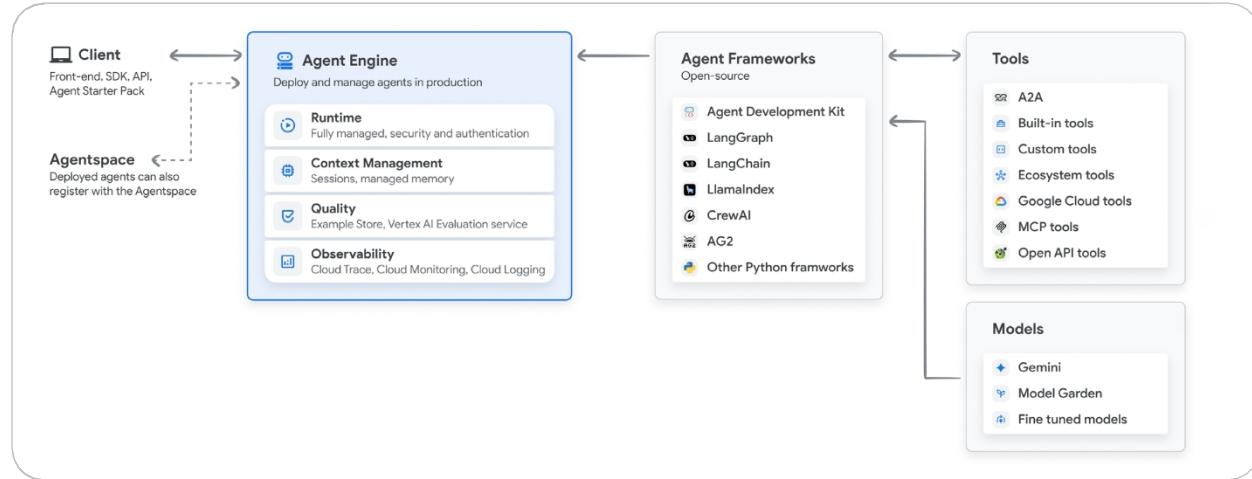


Рисунок 4: Конструктор Vefieox AI Agent из
<https://cloud.google.com/vefieox-ai/generative-ai/docs/agent-engine/overview>

Если вы не являетесь разработчиком программного обеспечения и экспертом DevOps, процесс развертывания вашего первого агента может показаться сложной задачей. Многие фреймворки агентов упрощают эту задачу с помощью команды [развертывания](#) или специальной платформы для развертывания агента, и их следует использовать для раннего изучения и внедрения. Переход к безопасной и готовой к производству среде обычно требует больших затрат времени и применения передовых методов, включая CI/CD и автоматическое тестирование [ваших агентов](#)²³.

Agent Ops: структурированный подход к непредсказуемому

При создании первых агентов вы будете вручную тестировать их поведение снова и снова. Когда вы добавляете функцию, работает ли она? Когда вы исправляете ошибку, не вызывает ли это другую проблему? Тестирование — это нормальный процесс при разработке программного обеспечения, но в случае с генеративным ИИ он работает по-другому.

Переход от традиционного детерминированного программного обеспечения к стохастическим агентным системам требует новой философии эксплуатации. Традиционные модульные тесты программного обеспечения могли просто оценивать `output == expected` (`выход == ожидаемый`), но это не работает, когда ответ агента по своему дизайну является вероятностным. Кроме того, поскольку язык сложен, для оценки «качества» обычно требуется LM — чтобы ответ агента делал все, что должен, ничего лишнего и с правильным тоном.

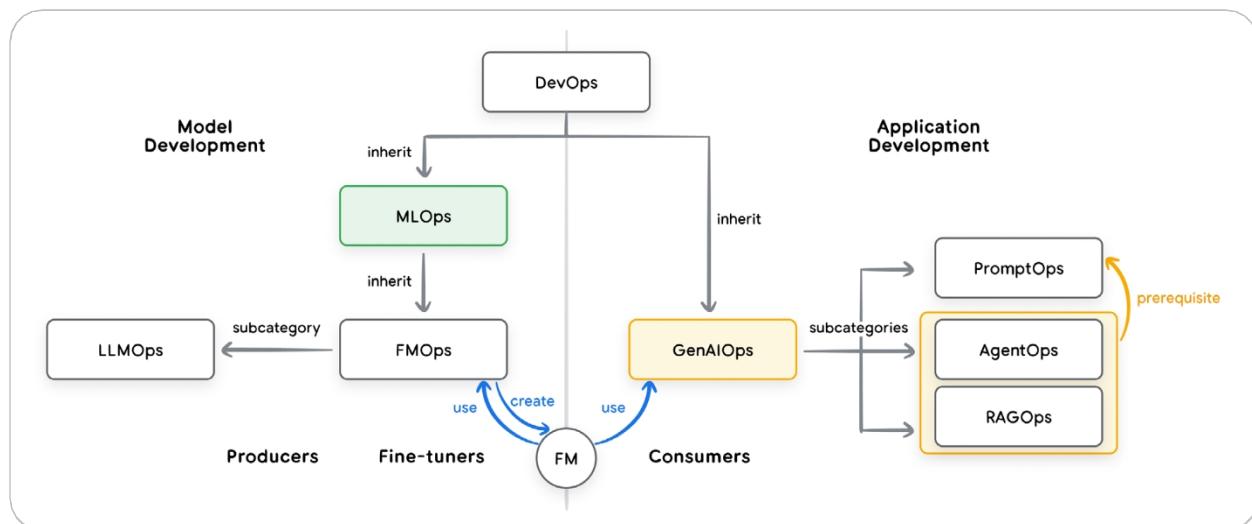


Рисунок 5: Взаимосвязь между операционными областями DevOps, MLOps и GenAIOps из <https://medium.com/@sokratis.kafiakis/genai-in-production-mlops-or-genaiops-25691c9becd0>

Agent Ops — это дисциплинированный, структурированный подход к управлению этой новой реальностью. Это естественная эволюция DevOps и MLOps, адаптированная к уникальным задачам создания, развертывания и управления агентами ИИ, превращающая непредсказуемость из недостатка в управляемую, измеримую и надежную функцию.²⁴ Для более полного и глубокого изучения см. технический документ, посвященный качеству агентов, в этой серии.

Измеряйте то, что важно: инструментируйте успех как А/В-эксперимент

Прежде чем улучшать своего агента, вы должны определить, что означает «лучше» в контексте вашего бизнеса. Сформулируйте свою стратегию наблюдаемости как А/В-тест и задайте себе вопрос: какие ключевые показатели эффективности (KPI) доказывают, что агент приносит пользу? Эти показатели должны выходить за рамки технической корректности и измерять реальное влияние: коэффициент достижения целей, оценки удовлетворенности пользователей, задержки выполнения задач, операционные затраты на одно взаимодействие и, что наиболее важно, влияние на бизнес-цели, такие как выручка, конверсия или удержание клиентов.

Этот обзор сверху вниз будет служить ориентиром для остальной части тестирования, направит вас на путь разработки, основанной на метриках, и позволит рассчитать рентабельность инвестиций.

Качество вместо «прошел/не прошел»: использование LM Judge

Бизнес-метрики не показывают, правильно ли ведет себя агент. Поскольку простое прохождение/непрохождение невозможно, мы переходим к оценке качества с помощью «LM в качестве судьи». Это предполагает использование мощной модели для оценки результатов работы агента по заранее определенной шкале: дал ли он правильный ответ? Был ли ответ основан на фактах? Следовал ли он инструкциям? Эта автоматизированная оценка, выполняемая на основе золотого набора данных подсказок, обеспечивает последовательную оценку качества.

Создание наборов данных для оценки, которые включают идеальные (или «золотые») вопросы и правильные ответы, может быть утомительным процессом. Для их создания необходимо отобрать сценарии из существующих производственных или разработческих взаимодействий с агентом. Набор данных должен охватывать весь спектр вариантов использования, с которыми, как вы ожидаете, будут сталкиваться ваши пользователи, а также несколько неожиданных. Хотя инвестиции в оценку быстро окупаются, результаты оценки всегда должны

проверяться экспертом в данной области, прежде чем их можно будет считать достоверными. Все чаще курирование и ведение этих оценок становится ключевой обязанностью менеджеров по продукту при поддержке экспертов в данной области.

Разработка на основе метрик: ваше решение о внедрении

После автоматизации десятков сценариев оценки и установления надежных показателей качества вы можете с уверенностью тестировать изменения в вашем агенте разработки. Процесс прост: запустите новую версию на всем наборе данных оценки и напрямую сравните ее показатели с существующей производственной версией. Эта надежная система исключает догадки, обеспечивая вам уверенность в каждом развертывании. Хотя автоматизированные оценки имеют решающее значение, не забывайте о других важных факторах, таких как задержка, стоимость и коэффициент успешности задач. Для максимальной безопасности используйте А/В-развертывания, чтобы постепенно внедрять новые версии и сравнивать эти реальные производственные показатели с результатами моделирования.

Отладка с помощью трассировок OpenTelemetry: ответ на вопрос «почему?».

Когда ваши показатели падают или пользователь сообщает об ошибке, вам нужно понять «почему». Трассировка OpenTelemetry — это высокоточная пошаговая запись всего пути выполнения агента (траектории), позволяющая отлаживать [шаги агента](#).²⁵ С помощью трассировок вы можете увидеть точный запрос, отправленный модели, внутреннее обоснование модели (если доступно), конкретный инструмент, который он выбрал для вызова, точные параметры, которые он сгенерировал для этого инструмента, и исходные данные, которые были возвращены в качестве наблюдения. Трассировки могут показаться сложными при первом взгляде, но они предоставляют детали, необходимые для диагностики и устранения первопричины любой проблемы. Незначительные детали трассировки могут быть преобразованы в метрики, но просмотр трассировок в первую очередь предназначен для отладки, а не

обзоры производительности. Данные трассировки могут быть легко собраны на таких платформах, как **Google Cloud Trace**, которые визуализируют и осуществляют поиск по огромному количеству трассировок, упрощая анализ первопричин.

Цените отзывы людей: руководство по автоматизации

Отзывы людей — это не раздражающий фактор, с которым нужно бороться, а самый ценный и богатый данными ресурс, который у вас есть для улучшения вашего агента. Когда пользователь подает отчет об ошибке или нажимает кнопку «не нравится», он дарит вам подарок: новый реальный крайний случай, который был упущен вашими автоматизированными сценариями оценки. Сбор и агрегирование этих данных имеют решающее значение; когда вы видите статистически значимое количество похожих отчетов или падений показателей, вы должны связать эти события с вашей аналитической платформой, чтобы сгенерировать аналитические данные и запустить alefis для операционных проблем. Эффективный процесс Agent Ops «замыкает цикл», собирая эту обратную связь, воспроизводя проблему и преобразуя этот конкретный сценарий в новый постоянный тестовый случай в вашем наборе данных оценки. Это гарантирует, что вы не только исправите ошибку, но и защитите систему от повторного возникновения всего класса ошибок.

Взаимодействие агентов

После создания высококачественных агентов вы захотите соединить их с пользователями и другими агентами. В нашей аналогии с телом человека это будет лицо агента. Существует разница между подключением к агентам и подключением агентов к данным и API; [агенты не являются инструментами](#)²⁶. Давайте предположим, что у вас уже есть инструменты, подключенные к вашим агентам, и теперь рассмотрим, как вы можете внедрить своих агентов в более широкую экосистему.

Агенты и люди

Наиболее распространенной формой взаимодействия агента и человека является пользовательский интерфейс. В своей простейшей форме это чат-бот, в котором пользователь вводит запрос, а агент, действующий в качестве бэкэнд-сервиса, обрабатывает его и возвращает блок текста. Более продвинутые агенты могут предоставлять структурированные данные, такие как JSON, для обеспечения богатых и динамичных интерфейсов. Взаимодействие человека в цикле (HITL) включают уточнение намерения, расширение цели, подтверждение и запросы на разъяснение.

Использование компьютера — это категория инструментов, в которой LM контролирует пользовательский интерфейс, часто с участием человека и под его контролем. Агент с поддержкой использования компьютера может решить, что следующим лучшим действием будет переход на новую страницу, выделение определенной кнопки или предварительное заполнение формы соответствующей [информацией](#)²⁷.

Вместо того, чтобы агент использовал интерфейс от имени пользователя, LM может изменить пользовательский интерфейс в соответствии с текущими потребностями. Это можно сделать с помощью инструментов, которые управляют пользовательским интерфейсом ([MCP UI](#))²⁸, или специализированных систем обмена сообщениями пользовательского интерфейса, которые могут синхронизировать состояние клиента с агентом ([AG UI](#))²⁹, и даже с помощью генерации индивидуальных интерфейсов ([A2UI](#))³⁰.

Конечно, человеческое взаимодействие не ограничивается экранами и клавиатурами. Передовые агенты преодолевают текстовый барьер и переходят к мультимодальному общению в режиме реального времени с «живым режимом», создавая более естественную, похожую на человеческую связь. Такие технологии, как [Gemini Live API](#)³¹, обеспечивают двунаправленную потоковую передачу, позволяя пользователю разговаривать с агентом и прерывать его, как в естественном разговоре.

Эта возможность коренным образом меняет характер сотрудничества между агентом и человеком. Имея доступ к камере и микрофону устройства, агент может видеть то, что видит пользователь, и слышать то, что он говорит, отвечая сгенерированной речью с задержкой, имитирующей человеческий разговор.

Это открывает широкий спектр возможностей, которые просто невозможны с текстом, от технического специалиста, получающего инструкции в режиме громкой связи во время ремонта оборудования, до покупателя, получающего советы по стилю в режиме реального времени. Это делает агента более интуитивным и доступным помощником.

Агенты и агенты

Так же как агенты должны связываться с людьми, они должны связываться друг с другом. По мере расширения использования ИИ в предприятии различные команды будут создавать разных специализированных агентов. Без общего стандарта для их связи потребовалось бы создать запутанную сеть хрупких, настраиваемых API-интеграций, которые невозможно поддерживать. Основная задача состоит из двух частей: обнаружение (как мой агент находит других агентов и узнает, что они могут делать?) и коммуникация (как мы обеспечим, чтобы они говорили на одном языке?).

Протокол Agent2Agent (A2A) — это открытый стандарт, разработанный для решения этой проблемы. Он действует как универсальный протокол обмена данными для агентской экономики. A2A позволяет любому агенту публиковать цифровую «визитную карточку», известную как Agent Card. Этот простой файл JSON содержит информацию о возможностях агента, его сетевом конечной точке и учетных данных безопасности, необходимых для взаимодействия с ним. Это упрощает и стандартизирует процесс обнаружения. В отличие от MCP, который фокусируется на решении транзакционных запросов, коммуникация Agent 2 Agent обычно предназначена для решения дополнительных задач.

После обнаружения агенты общаются с помощью архитектуры, ориентированной на задачи. Вместо простого запроса-ответа взаимодействия оформляются как асинхронные «задачи». Клиентский агент отправляет запрос на задачу серверному агенту, который затем может предоставлять потоковые обновления, работая над проблемой через долгосрочное соединение. Этот надежный, стандартизованный протокол связи является последним элементом головоломки, позволяющим создать совместные мультиагентные системы уровня 3, которые представляют собой передовую технологию автоматизации. A2A превращает набор изолированных агентов в настоящую, взаимодействующую экосистему.

Агенты и деньги

Поскольку агенты ИИ выполняют за нас все больше задач, некоторые из них связаны с покупкой или продажей, ведением переговоров или содействием в проведении транзакций. Современный Интернет создан для людей, которые нажимают кнопку «купить», и ответственность лежит на человеке. Если автономный агент нажимает кнопку «купить», это создает кризис доверия — если что-то пойдет не так, кто будет виноват? Это сложные вопросы авторизации, подлинности и ответственности. Чтобы раскрыть потенциал настоящей агентской экономики, нам нужны новые стандарты, которые позволят агентам безопасно и надежно совершать транзакции от имени своих пользователей.

Эта новая область еще далека от стабильности, но два ключевых протокола прокладывают путь к ее развитию. **Agent Payments Protocol (AP2)** — это открытый протокол, разработанный как универсальный язык для агентской коммерции. Он расширяет протоколы типа A2A за счет введения криптографически подписанных цифровых «мандатов». Они служат в качестве проверяемого доказательства намерения пользователя, создавая невозвратный контрольный след для каждой транзакции. Это позволяет агенту безопасно просматривать, вести переговоры и совершать транзакции в глобальном масштабе на основании полномочий, делегированных пользователем. Дополняет его **x402** — открытый интернет-протокол платежей, использующий стандартный статус-код HTTP 402 «Требуется оплата». Он обеспечивает беспрепятственные микроплатежи между машинами, позволяя агенту оплачивать такие вещи, как доступ к API или цифровой контент, на основе оплаты по факту использования, без необходимости создания сложных учетных записей или подписок. Вместе эти протоколы создают основополагающий уровень доверия для агентского веба.

Обеспечение безопасности единственного агента: компромисс между доверием и безопасностью

Когда вы создаете своего первого агента ИИ, вы сразу сталкиваетесь с фундаментальным противоречием: компромисс между полезностью и безопасностью. Чтобы агент был полезен, вы должны наделить его полномочиями — автономией в принятии решений и инструментами для выполнения таких действий, как отправка электронных писем или запрос данных из баз данных. Однако каждое предоставленное полномочие сопряжено с соответствующим риском. Основные проблемы безопасности связаны с **несанкционированными действиями** — непреднамеренными или вредными поступками —

и раскрытие конфиденциальных данных. Вы хотите дать своему агенту достаточно свободы, чтобы он мог выполнять свою работу, но в то же время ограничить его, чтобы он не попал в опасную ситуацию, особенно если эта ситуация связана с необратимыми действиями или конфиденциальными данными вашей компании.³²

Для управления этим вы не можете полагаться исключительно на суждения модели ИИ, поскольку они могут быть манипулированы с помощью таких техник, как [вставка подсказок](#)³³. Вместо этого лучшей практикой является гибридный [подход глубокой защиты](#)⁽³⁴⁾. Первый уровень состоит из **традиционных детерминированных** ограждений — набора жестко запрограммированных правил, которые действуют как узкие места безопасности вне рамок рассуждений модели. Это может быть механизм политики, который блокирует любые покупки на сумму более 100 долларов или требует явного подтверждения пользователя, прежде чем агент сможет взаимодействовать с внешним API. Этот уровень обеспечивает предсказуемые и поддающиеся аудиту жесткие ограничения на возможности агента.

Второй уровень использует **защиту на основе логического мышления**, применения ИИ для обеспечения безопасности ИИ. Это включает в себя обучение модели быть более устойчивой к атакам (противостоящее обучение) и использование более мелких, специализированных «моделей охраны», которые действуют как аналитики по безопасности. Эти модели могут проверять предлагаемый агентством план перед его выполнением, отмечая потенциально рискованные или нарушающие политику шаги для пересмотра. Эта гибридная модель, сочетающая в себе жесткую определенность кода с контекстной осведомленностью ИИ, создает надежную систему безопасности даже для одного агента, гарантируя, что его мощность всегда соответствует его назначению.

Идентичность агента: новый класс принципала

В традиционной модели безопасности есть человеческие пользователи, которые могут использовать OAuth или SSO, и есть службы, которые используют IAM или служебные учетные записи. Агенты добавляют третью категорию принципалов. Агент — это не просто кусок кода; это автономный участник, новый вид *принципала*, который требует собственной проверяемой идентичности. Так же, как сотрудникам выдают идентификационные бейджи, каждому агенту на платформе должен быть выдан безопасный, поддающийся проверке «цифровой паспорт». Этот агент

Идентичность отличается от идентичности пользователя, который ее вызвал, и разработчика, который ее создал. Это фундаментальное изменение в подходе к управлению идентичностью и доступом (IAM) в предприятии.

Проверка каждой идентичности и контроль доступа для всех из них являются основой безопасности агента. Как только агент получает криптографически проверяемую идентичность (часто с использованием стандартов, таких как [SPIFFE](#)³⁵), ему могут быть предоставлены собственные специфические права с минимальными привилегиями. `SalesAgent` получает доступ на чтение/запись к CRM, в то время как `HRonboardingAgent` явно лишается такого доступа. Такой детальный контроль имеет решающее значение. Он гарантирует, что даже в случае компрометации одного агента или его непредвиденного поведения потенциальный радиус поражения будет ограничен. Без конструкции идентичности агента агенты не могут работать от имени людей с ограниченными делегированными полномочиями.

Основная сущность	Аутентификация / Верификация	Примечания
Пользователи	Аутентифицированы с помощью OAuth или SSO	Люди, обладающие полной автономией и ответственностью за свои действия
Агенты (новая категория принципов)	Проверенные с помощью SPIFFE	Агенты наделены полномочиями и действуют от имени пользователей
Служебные учетные записи	Интегрированы в IAM	Приложения и контейнеры, полностью детерминированные, не ответственность за действия

Таблица 1: Неполный пример различных категорий участников для аутентификации

Политики ограничения доступа

Политика — это форма авторизации (AuthZ), отличная от аутентификации (AuthN). Как правило, политики ограничивают возможности субъекта; например, «Пользователи из отдела маркетинга могут получить доступ только к этим 27 конечным точкам API и не могут выполнять команды DELETE». При разработке агентов нам необходимо применять разрешения к агентам, их инструментам, другим внутренним агентам, контексту, которым они могут делиться, и удаленным агентам. Подумайте об этом так: если вы добавляете все API, данные, инструменты и агенты в свою систему, то вы должны ограничить доступ только к тому поднабору возможностей, который необходим для выполнения их задач. Рекомендуемый подход: применять принцип минимальных привилегий, оставаясь [контекстуально релевантным](#).⁽³⁶⁾

Защита агента ADK

После установления основных принципов идентификации и политики обеспечение безопасности агента, созданного с помощью Agent Development Kit (ADK), становится практическим упражнением по применению этих концепций через код и [конфигурацию](#).⁽³⁷⁾

Как описано выше, этот процесс требует четкого определения идентичностей: учетной записи пользователя (например, OAuth), учетной записи службы (для запуска кода), идентичности агента (для использования делегированных полномочий). После аутентификации следующий уровень защиты включает установление политик для ограничения доступа к службам. Это часто делается на уровне управления API, наряду с управлением, поддерживающим службы MCP и A2A.

Следующий уровень — это встраивание защитных механизмов в ваши инструменты, модели и субагенты для обеспечения соблюдения политик. Это гарантирует, что независимо от причин LM или того, что может подсказывать злонамеренный запрос, собственная логика инструмента откажется выполнять небезопасные или несоответствующие политике действия. Такой подход обеспечивает предсказуемую и поддающуюся аудиту базовую линию безопасности, преобразуя абстрактные политики безопасности в [конкретный, надежный код](#).⁽³⁸⁾

Для более динамичной безопасности, которая может адаптироваться к поведению агента во время выполнения, ADK предоставляет **обратные вызовы и плагины**. `Before_tool_callback` позволяет проверять параметры вызова инструмента перед его запуском, сопоставляя их с текущим состоянием агента, чтобы предотвратить несогласованные действия. Для более повторно используемых политик вы можете создавать плагины. Распространенным шаблоном является «Gemini as a Judge»⁽³⁹⁾, который использует быструю и недорогую модель, такую как Gemini Flash-Lite, или вашу собственную точно настроенную модель Gemma для проверки ввода пользователя и вывода агента на предмет быстрых вставок или вредоносного контента в режиме реального времени.

Для организаций, которые предпочитают полностью управляемое решение корпоративного уровня для этих динамических проверок, **Model Armor** может быть интегрирован в качестве дополнительной услуги. Model Armor действует как специализированный уровень безопасности, который проверяет запросы и ответы на наличие широкого спектра угроз, включая внедрение запросов, попытки джейлбрейка, утечку конфиденциальных данных (PII) и [вредоносные URL-адреса](#)⁴⁰. Передав эти сложные задачи по обеспечению безопасности специальной службе, разработчики могут обеспечить постоянную и надежную защиту, не создавая и не поддерживая эти защитные механизмы самостоятельно. Этот гибридный подход в рамках ADK, сочетающий в себе надежную идентификацию, детерминированную логику в инструменте, динамические защитные механизмы на базе искусственного интеллекта и дополнительные управляемые службы, такие как Model Armor, позволяет создать единый агент, который является одновременно мощным и надежным.

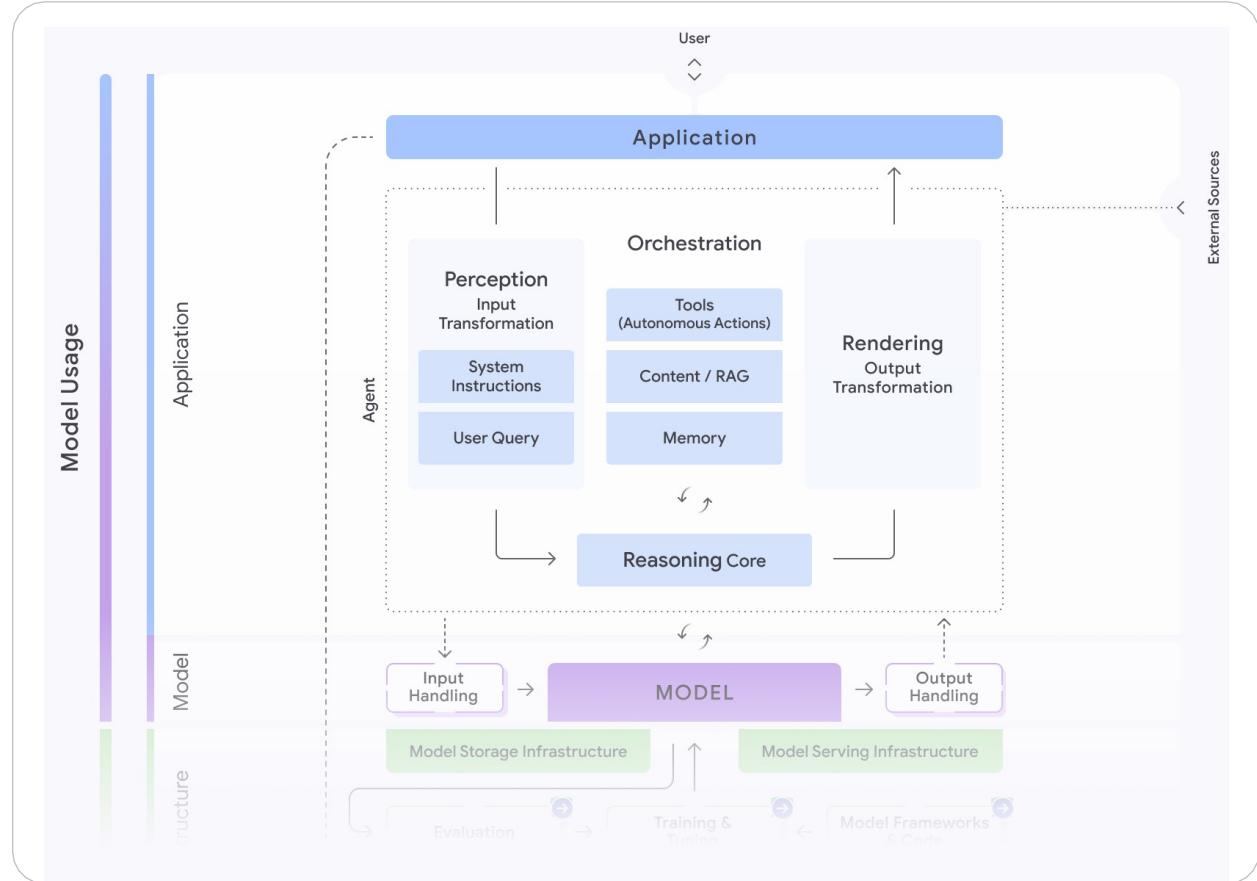


Рисунок 6: Безопасность и агенты из <https://saif.google/focus-on-agents>

Масштабирование от одного агента до корпоративного парка

Успех производства одного агента ИИ — это триумф. Масштабирование до парка из сотен агентов — это вызов для архитектуры. Если вы создаете одного или двух агентов, ваши заботы в первую очередь касаются безопасности. Если вы создаете много агентов, вы должны проектировать системы, способные обрабатывать гораздо больше. Так же, как и в случае с разрастанием API, когда агенты и инструменты распространяются по всей организации,

они создают новую сложную сеть взаимодействий, потоков данных и потенциальных уязвимостей безопасности. Для управления этой сложностью требуется уровень управления более высокого порядка, интегрирующий все ваши идентификаторы и политики и перенаправляющий их в центральную плоскость управления.

Безопасность и конфиденциальность: укрепление границ агентов

Платформа корпоративного уровня должна решать уникальные проблемы безопасности и конфиденциальности, присущие генеративному ИИ, даже если работает только один агент. Сам агент становится новым вектором атаки. Злоумышленники могут попытаться **внедрить промпт-инъекцию**, чтобы перехватить инструкции агента, или **отправиться данные**, чтобы повредить информацию, которую он использует для обучения или RAG. Кроме того, плохо ограниченный агент может непреднамеренно раскрыть конфиденциальные данные клиентов или проприетарную информацию в своих ответах.

Надежная платформа обеспечивает стратегию глубокой защиты для снижения этих рисков. Она работает с данными, гарантируя, что конфиденциальная информация предприятия никогда не будет использоваться для обучения базовых моделей и будет защищена с помощью таких средств контроля, как VPC Service Controls. Она требует фильтрации входных и выходных данных, действуя как брандмауэр для запросов и ответов. Наконец, платформа должна предлагать договорные меры защиты, такие как компенсация за ущерб интеллектуальной собственности, как для обучающих данных, так и для сгенерированных результатов, что дает предприятиям юридическую и техническую уверенность, необходимую для развертывания агентов в производственной среде.

Управление агентами: плоскость управления вместо разрастания

По мере распространения агентов и их инструментов по всей организации они создают новую сложную сеть взаимодействий и потенциальных уязвимостей, что часто называют «разбросанностью агентов». Для управления этим процессом необходимо выйти за рамки обеспечения безопасности отдельных агентов и внедрить архитектурный подход более высокого уровня: центральный шлюз, который служит контрольной плоскостью для всей деятельности агентов.

Представьте себе оживленный мегаполис с тысячами автономных транспортных средств — пользователей, агентов и инструментов, — все из которых движутся с определенной целью. Без светофоров, номерных знаков и центральной системы управления царил бы хаос. Подход на основе шлюза создает такую систему управления, устанавливая обязательную точку входа для всего трафика агентов, включая запросы от пользователя к агенту или взаимодействия с пользовательским интерфейсом

, вызовы агента инструменту (через MCP), сотрудничество агента с агентом (через A2A) и прямые запросы на вывод к LM. Находясь на этом критическом перекрестке, организация может проверять, направлять, контролировать и управлять каждым взаимодействием.

Эта плоскость управления выполняет две основные взаимосвязанные функции:

- 1. Принудительное выполнение политик во время выполнения:** она действует как архитектурный узловый пункт для реализации безопасности. Она обрабатывает аутентификацию («Знаю ли я, кто этот участник?») и авторизацию («Имеют ли они разрешение на это?»). Централизованное принудительное выполнение обеспечивает «единое окно» для наблюдаемости, создавая общие журналы, метрики и трассировки для каждой транзакции. Это превращает беспорядочную смесь разнородных агентов и рабочих процессов в прозрачную и поддающуюся аудиту систему.
- 2. Централизованное управление:** для эффективного обеспечения соблюдения политик шлюз необходим источник достоверной информации. Его обеспечивает центральный реестр — корпоративный магазин приложений для агентов и инструментов. Этот реестр позволяет разработчикам находить и повторно использовать существующие ресурсы, предотвращая избыточную работу, а администраторам — получать полный перечень ресурсов. Что еще более важно, он обеспечивает формальный жизненный цикл агентов и инструментов, позволяя проводить проверки безопасности перед публикацией, версионированием и созданием детальных политик, которые определяют, какие бизнес-подразделения могут получить доступ к каким агентам.

Объединяя шлюз выполнения с центральным реестром управления, организация преобразует риск хаотичного разрастания в управляемую, безопасную и эффективную экосистему.

Стоимость и надежность: инфраструктурная основа

В конечном итоге, агенты корпоративного уровня должны быть как надежными, так и экономически эффективными. Агент, который часто выходит из строя или предоставляет медленные результаты, имеет отрицательную рентабельность инвестиций. И наоборот, агент, который является непомерно дорогим, не может масштабироваться для удовлетворения потребностей бизнеса. Базовая
должна быть спроектирована таким образом, чтобы управлять этим компромиссом, обеспечивая безопасность и соблюдение нормативных требований и суверенитета данных.

В некоторых случаях вам нужна функция масштабирования до нуля, когда у вас нерегулярный трафик к определенному агенту или подфункции. Для критически важных рабочих нагрузок, чувствительных к задержкам,
платформа должна предлагать выделенную гарантированную емкость, такую как [Provisioned Throughput⁴¹](#) для служб LM или соглашения об уровне обслуживания (SLA) 99,9% для сред выполнения, таких как [Cloud Run⁴²](#). Это обеспечивает предсказуемую производительность, гарантируя, что ваши наиболее важные агенты всегда будут отзывчивыми, даже при высокой нагрузке. Предоставляя такой спектр инфраструктурных опций в сочетании с комплексным мониторингом как затрат, так и производительности, вы создаете окончательную, необходимую основу для масштабирования агентов ИИ от многообещающей инновации до основного, надежного компонента предприятия.

Как агенты развиваются и учатся

Агенты, развернутые в реальном мире, работают в динамичных средах, где политики, технологии и форматы данных постоянно меняются. Без способности адаптироваться производительность агента со временем снижается — этот процесс часто называют «старением» — что приводит к потере полезности и доверия. Ручное обновление большого парка агентов для того, чтобы идти в ногу с этими изменениями, неэкономично и медленно. Более масштабируемым решением является разработка агентов, которые могут самостоятельно учиться и развиваться, улучшая качество своей работы с [минимальными инженерными усилиями](#) (⁴³).

Как агенты учатся и развиваются самостоятельно

Подобно людям, агенты учатся на опыте и внешних сигналах. Этот процесс обучения подпитывается несколькими источниками информации:

- **Опыт выполнения:** агенты учатся на основе фактов выполнения, таких как журналы сеансов, трассировки и память, которые фиксируют успехи, неудачи, взаимодействия с инструментами и траектории принятия решений.
. Важно отметить, что сюда входит обратная связь Human-in-the-Loop (HITL), которая обеспечивает авторитетные исправления и рекомендации.
- **Внешние сигналы:** обучение также стимулируется новыми внешними документами, такими как обновленные корпоративные политики, публичные нормативные рекомендации или критика со стороны других агентов.

Эта информация затем используется для оптимизации будущего поведения агента. Вместо простого обобщения прошлых взаимодействий, передовые системы создают обобщаемые афифакты для руководства будущими задачами. Наиболее успешные методы адаптации делятся на две категории:

- **Улучшенная контекстная инженерия:** система постоянно совершенствует свои подсказки, примеры с несколькими выстрелами и информацию, которую она извлекает из памяти. Оптимизируя контекст, предоставляемый LM для каждой задачи, она увеличивает вероятность успеха.
- **Оптимизация и создание инструментов:** Рассуждения агента позволяют выявлять пробелы в его возможностях и принимать меры для их устранения. Это может включать в себя получение доступа к новому инструменту, создание нового инструмента на лету (например, скрипта Python) или модификацию существующего инструмента (например, обновление схемы API).

Дополнительные методы оптимизации, такие как динамическая переконфигурация многоагентных шаблонов проектирования или использование метода укрепления обучения на основе обратной связи от человека (RLHF), являются активными областями исследований.

Пример: изучение новых правил соответствия

Рассмотрим корпоративного агента, работающего в строго регулируемой отрасли, такой как финансы или биологические науки. Его задача — генерировать отчеты, которые должны соответствовать правилам конфиденциальности и нормативным требованиям (например, GDPR).

Это можно реализовать с помощью многоагентного рабочего процесса:

1. **Агент запросов** извлекает необработанные данные в ответ на запрос пользователя.
2. **Агент по созданию отчетов** синтезирует эти данные в черновой вариант отчета.
3. **Агент-критик**, вооруженный известными правилами соответствия, проверяет отчет. Если он сталкивается с неоднозначностью или требует окончательного утверждения, он передает его человеку-эксперту в данной области.
4. **Агент-обучающий** наблюдает за всем взаимодействием, уделяя особое внимание корректирующей обратной связи от эксперта-человека. Затем он обобщает эту обратную связь в новое, повторно используемое руководство (например, обновленное правило для агента-критика или уточненный контекст для агента-репофи).

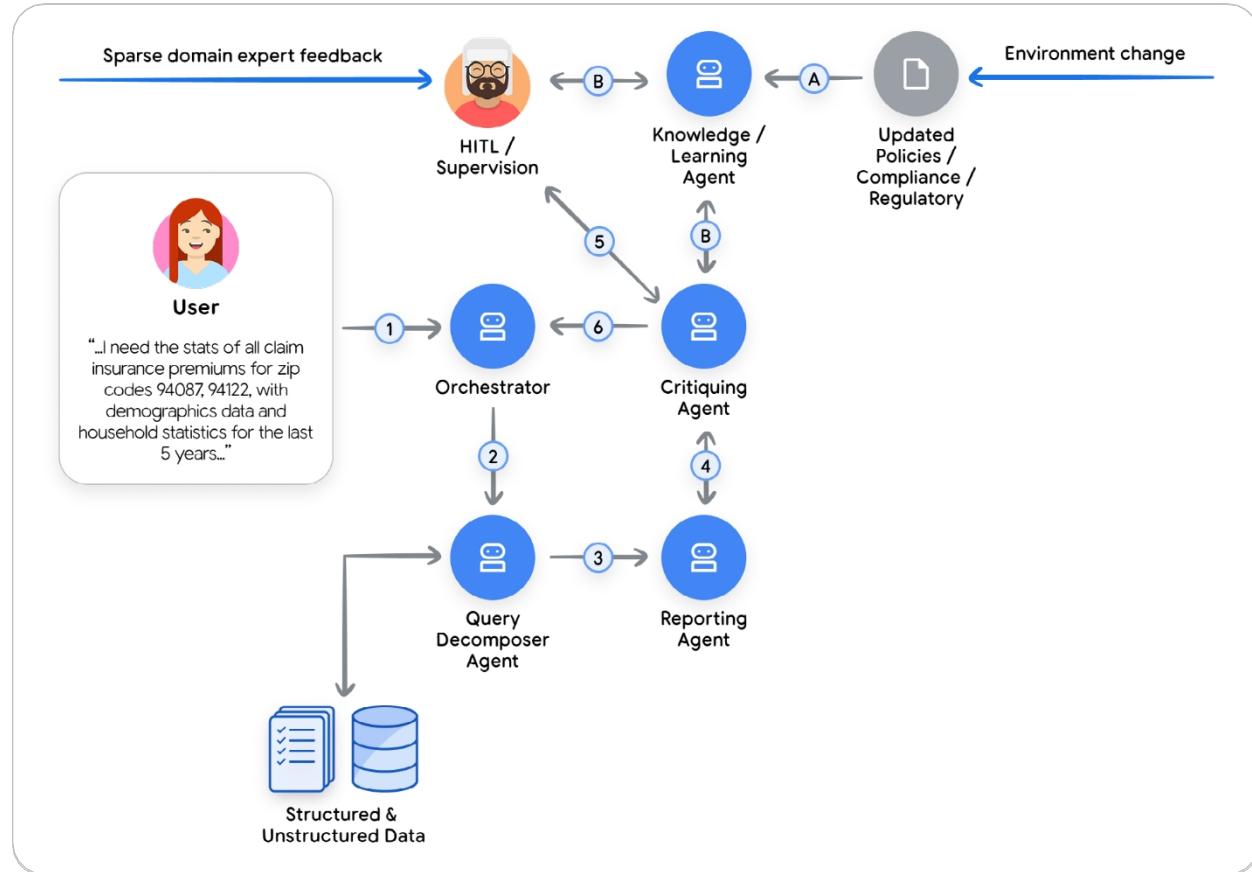


Рисунок 7: Пример многоагентного рабочего процесса для руководящих принципов соответствия

Например, если эксперт-человек отмечает, что статистические данные о домохозяйствах должны быть анонимизированы, обучающийся агент записывает эту поправку. В следующий раз, когда будет сгенерирован аналогичный отчет, агент-критик автоматически применит это новое правило, что снизит необходимость вмешательства человека. Этот цикл критики, обратной связи от человека и обобщения позволяет системе автономно адаптироваться к меняющимся требованиям соответствия⁴⁴.

Моделирование и Agent Gym — следующий рубеж

Представленный нами шаблон проектирования можно отнести к категории онлайн-обучения, при котором агенты должны учиться с помощью ресурсов и шаблонов проектирования, с которыми они были разработаны. В настоящее время

В настоящее время исследуются более продвинутые подходы, в которых используется специальная платформа, разработанная для оптимизации мультиагентной системы в автономных процессах с помощью продвинутых инструментов и возможностей, которые не являются частью мультиагентной среды выполнения. Ключевыми характеристиками такого [Agent Gym⁴⁵](#) являются:

1. Он не находится в пути выполнения. Это автономная внепроизводственная платформа, поэтому она может использовать любую модель LM, автономные инструменты, облачные приложения и многое другое
2. Она предлагает среду моделирования, поэтому агент может «тренироваться» на новых данных и учиться. Эта среда моделирования отлично подходит для «метода проб и ошибок» с множеством путей оптимизации.
3. Она может вызывать передовые генераторы синтетических данных, которые направляют симуляцию, чтобы она была максимально реалистичной, и подвергают агента испытанию под давлением (это может включать передовые техники, такие как красная команда, динамическая оценка и семейство критикующих агентов).
4. Арсенал инструментов оптимизации не является фиксированным, и он может принимать новые инструменты (либо через открытые протоколы, такие как MCP или A2A), либо в более продвинутой настройке - изучать новые концепции и создавать инструменты вокруг них.
5. Наконец, даже такие конструкции, как Agent Gym, могут оказаться неспособными преодолеть крайний случай *сезиайн* (из-за хорошо известной проблемы «племенного знания» в предприятии). В таких случаях мы видим, что Agent Gym способен подключаться к человеческой структуре доменных экспертов и консультироваться с ними по поводу правильного набора результатов, которые будут служить ориентиром для следующего набора оптимизаций.

Примеры продвинутых агентов

Google Co-Scientist

Co-Scientist — это продвинутый агент искусственного интеллекта, разработанный для работы в качестве виртуального научного сотрудника, ускоряющего научные открытия путем систематического изучения сложных проблемных пространств. Он позволяет исследователям определять цель, основываясь на указанных общедоступных и проприетарных источниках знаний, а затем генерировать и оценивать ландшафт новых гипотез.

Для достижения этой цели Co-Scientist создает целую экосистему агентов, сотрудничающих друг с другом.

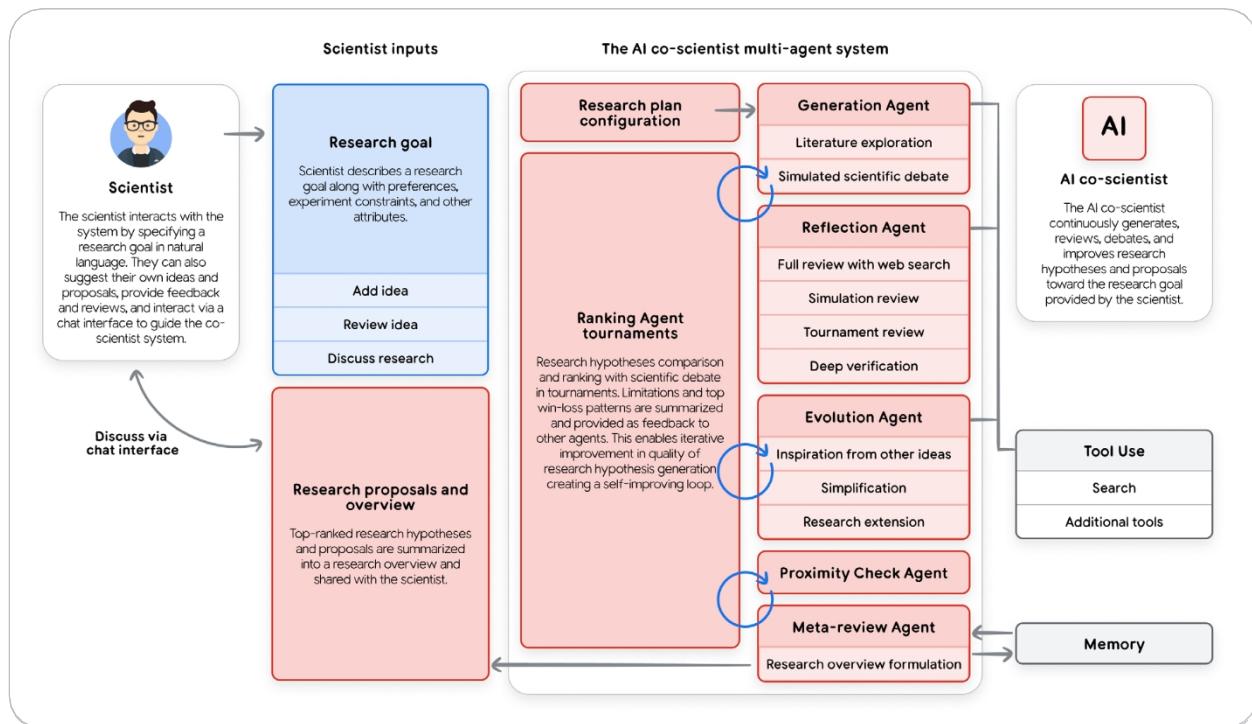


Рисунок 8: Система проектирования AI co-scientist

Представьте себе эту систему как менеджера исследовательского проекта. Сначала ИИ берет общую исследовательскую цель и создает подробный план проекта. Затем агент «Супервайзер» выступает в роли менеджера, делегируя задачи команде специализированных агентов и распределяя ресурсы, такие как вычислительная мощность. Такая структура обеспечивает легкое масштабирование проекта и совершенствование методов работы команды по мере продвижения к конечной цели.

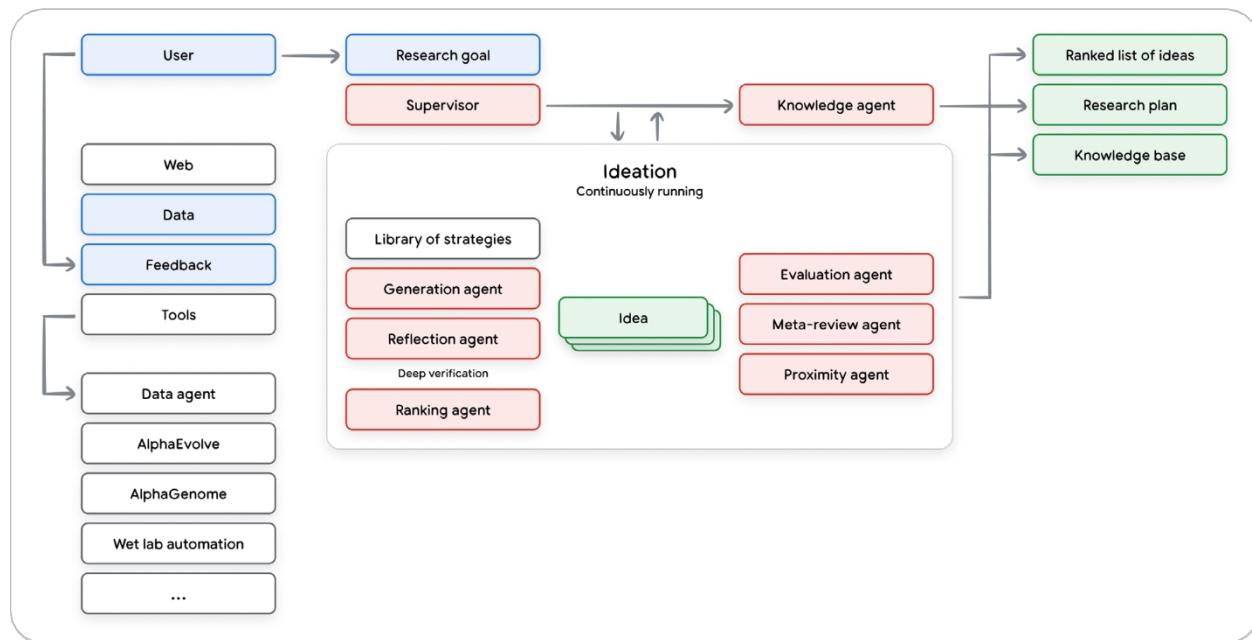


Рисунок 9: Рабочий процесс со-ученого с несколькими агентами

Различные агенты работают в течение нескольких часов или даже дней, постоянно совершенствуя сгенерированные гипотезы, запуская циклы и метацикли, которые улучшают не только сгенерированные идеи, но и способ, которым мы оцениваем и создаем новые идеи.

Агент AlphaEvolve

Еще одним примером продвинутой системы агентов является AlphaEvolve, агент искусственного интеллекта, который обнаруживает и оптимизирует алгоритмы для сложных задач в математике и информатике.

AlphaEvolve работает, сочетая творческое генерирование кода наших языковых моделей Gemini с автоматизированной системой оценки. Он использует эволюционный процесс: ИИ генерирует потенциальные решения, оценщик оценивает их, и наиболее перспективные идеи используются в качестве вдохновения для следующего поколения кода.

Этот подход уже привел к значительным прорывам, в том числе:

- Повышение эффективности центров обработки данных Google, проектирования микросхем и обучения искусственного интеллекта.
- Открытие более быстрых алгоритмов умножения матриц.
- Поиск новых решений открытых математических задач.

AlphaEvolve превосходно справляется с задачами, в которых проверка качества решения гораздо проще, чем его поиск.

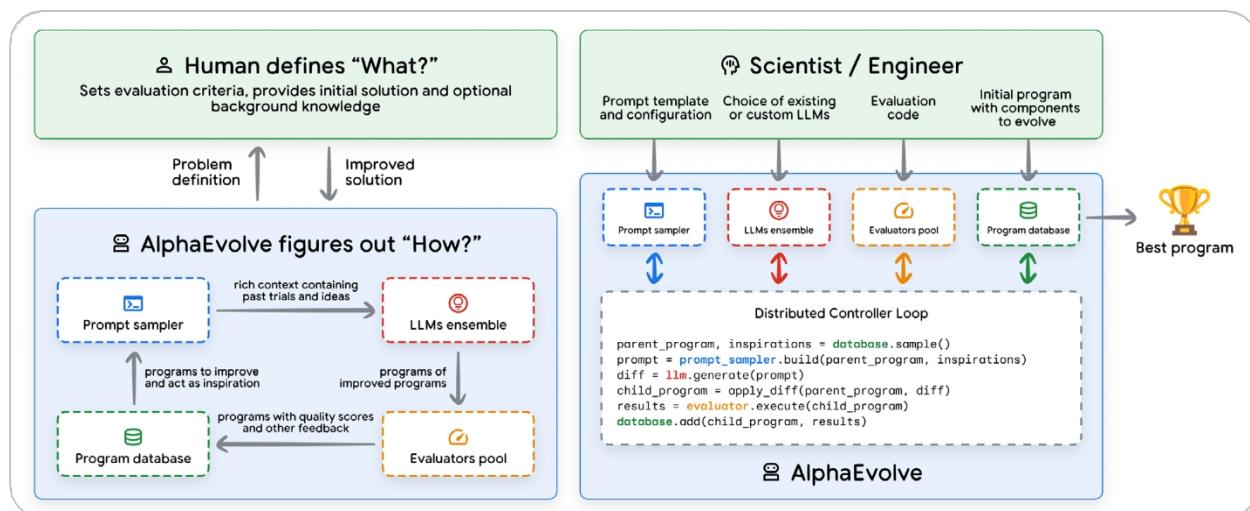


Рисунок 10: Система проектирования Alpha Evolve

AlphaEvolve разработана для глубокого, итеративного партнерства между людьми и ИИ. Это сотрудничество работает двумя основными способами:

- Прозрачные решения:** ИИ генерирует решения в виде кода, понятного человеку. Такая прозрачность позволяет пользователям понимать логику, получать новые знания, доверять результатам и напрямую изменять код в соответствии со своими потребностями.
- Экспертное руководство:** человеческий опыт необходим для определения проблемы. Пользователи направляют ИИ, уточняя метрики оценки и управляя исследованием, что предотвращает использование системой непреднамеренных лазеек в определении проблемы. Этот интерактивный цикл гарантирует, что окончательные решения будут одновременно мощными и практическими.

Результатом работы агента является постоянное совершенствование кода, который продолжает улучшать показатели, заданные человеком.

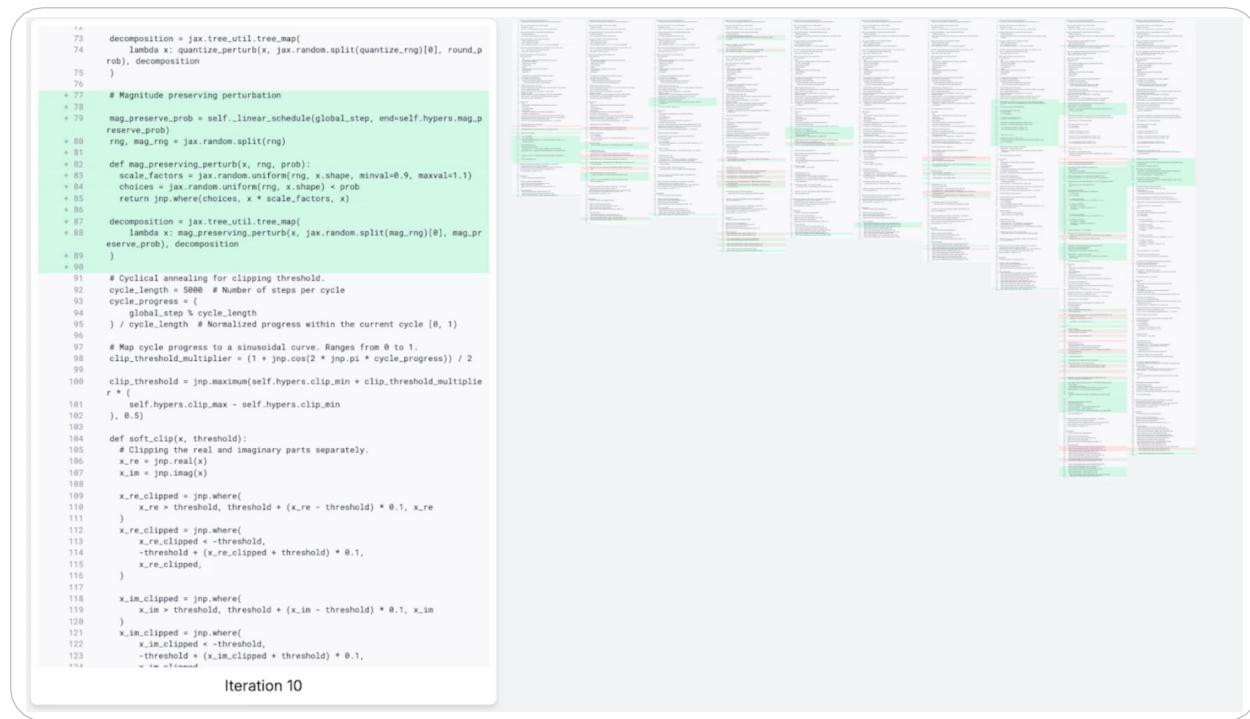


Рисунок 11: Эволюция алгоритма

Заключение

Генеративные ИИ-агенты знаменуют собой важную эволюцию, превращая искусственный интеллект из пассивного инструмента для создания контента в активный, автономный инструмент для решения проблем. В данном документе представлена формальная структура для понимания и построения этих систем, выходящая за рамки прототипа и позволяющая создать надежную архитектуру производственного уровня.

Мы разбили агент на три основных компонента: **модель рассуждений** («мозг»), **инструменты для действий** («руки») и управляющий **уровень оркестрации** («нервная система»). Именно бесшовная интеграция этих компонентов, работающих в непрерывном цикле «думай, действуй, наблюдай», раскрывает истинный потенциал агента. Классифицируя агентские системы — от уровня 1 «Подключенный решатель проблем» до уровня 3 «Совместная мультиагентная система» — архитекторы и руководители продуктов теперь могут стратегически определять масштаб своих амбиций в соответствии со сложностью поставленной задачи.

Основная задача и возможность заключаются в новой парадигме разработчика. Мы больше не являемся просто «каменщиками», определяющими явную логику; мы — «архитекторы» и «директора», которые должны направлять, ограничивать и отлаживать автономную сущность. Гибкость, которая делает LM настолько мощными, также является источником их ненадежности. Таким образом, успех заключается не только в начальном запросе, но и в инженерной строгости, примененной ко всей системе: в надежных контрактах инструментов, устойчивой обработке ошибок, сложной управлении контекстом и всесторонней оценке.

Принципы и архитектурные шаблоны, изложенные здесь, служат основополагающим планом. Они являются ориентирами для навигации по этой новой границе программного обеспечения, позволяя нам создавать не просто «автоматизацию рабочих процессов», а по-настоящему совместных, способных и адаптируемых новых членов наших команд. По мере развития этой технологии дисциплинированный архитектурный подход станет решающим фактором в использовании всего потенциала агентного ИИ.

Примечания

1. Джулия Визингер, Патрик Марлоу и др. 2024 «Агенты».
Доступно по адресу: [hflps://www.kaggle.com/whitepaper-agents](https://www.kaggle.com/whitepaper-agents).
2. Антонио Гулли, Лави Нигам и др. 2025 «Agents Companion».
Доступно по адресу: [hflps://www.kaggle.com/whitepaper-agent-companion](https://www.kaggle.com/whitepaper-agent-companion).
3. Шунью Яо, Ю. и др., 2022, «ReAct: Синергия рассуждений и действий в языковых моделях».
Доступно по адресу: [hflps://arxiv.org/abs/2210.03629](https://arxiv.org/abs/2210.03629).
4. Wei, J., Wang, X. et al., 2023, «Chain-of-Thought Prompting Elicits Reasoning in Large Language Models» (Цепочка мыслей вызывает рассуждения в больших языковых моделях). Доступно по адресу:
[hflps://arxiv.org/pdf/2201.11903.pdf](https://arxiv.org/pdf/2201.11903.pdf).
5. Шунью Яо, Ю. и др., 2022, «ReAct: синергия рассуждений и действий в языковых моделях».
Доступно по адресу: [hflps://arxiv.org/abs/2210.03629](https://arxiv.org/abs/2210.03629).
6. [hflps://www.amazon.com/Agentic-Design-Paflerns-Hands-Intelligent/dp/3032014018](https://www.amazon.com/Agentic-Design-Paflerns-Hands-Intelligent/dp/3032014018)
7. Shunyu Yao, et. al., 2024, «τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains» (τ-bench: тест для оценки взаимодействия между инструментом, агентом и пользователем в реальных условиях).
Доступно по адресу: [hflps://arxiv.org/abs/2406.12045](https://arxiv.org/abs/2406.12045).
8. [hflps://afiiificalanalysis.ai/guide](https://afiiificalanalysis.ai/guide)
9. [hflps://cloud.google.com/vefiex-ai/generative-ai/docs/model-reference/vefiex-ai-model-optimizer](https://cloud.google.com/vefiex-ai/generative-ai/docs/model-reference/vefiex-ai-model-optimizer)
10. [hflps://gemini.google/overview/gemini-live/](https://gemini.google/overview/gemini-live/)
11. [hflps://cloud.google.com/vision?e=48754805&hl=en](https://cloud.google.com/vision?e=48754805&hl=en)
12. [hflps://cloud.google.com/speech-to-text?e=48754805&hl=en](https://cloud.google.com/speech-to-text?e=48754805&hl=en)
13. [hflps://medium.com/google-cloud/genaiops-operationalize-generative-ai-a-practical-guide-d5bedaa59d78](https://medium.com/google-cloud/genaiops-operationalize-generative-ai-a-practical-guide-d5bedaa59d78)
14. [hflps://cloud.google.com/vefiex-ai/generative-ai/docs/agent-engine/code-execution/overview](https://cloud.google.com/vefiex-ai/generative-ai/docs/agent-engine/code-execution/overview)
15. [hflps://ai.google.dev/gemini-api/docs/function-calling](https://ai.google.dev/gemini-api/docs/function-calling)
16. [hflps://github.com/modelcontextprotocol/](https://github.com/modelcontextprotocol/)
17. [hflps://ai.google.dev/gemini-api/docs/google-search](https://ai.google.dev/gemini-api/docs/google-search)

18. [hflps://google.github.io/adk-docs/](https://google.github.io/adk-docs/)
19. [hflps://google.github.io/adk-docs/sessions/memory/](https://google.github.io/adk-docs/sessions/memory/)
20. [hflps://cloud.google.com/architecture/choose-design-paflern-agentic-ai-system](https://cloud.google.com/architecture/choose-design-paflern-agentic-ai-system)
21. [hflps://cloud.google.com/vefiex-ai/generative-ai/docs/agent-engine/overview](https://cloud.google.com/vefiex-ai/generative-ai/docs/agent-engine/overview)
22. [hflps://cloud.google.com/kubernetes-engine/docs/concepts/gke-and-cloud-run](https://cloud.google.com/kubernetes-engine/docs/concepts/gke-and-cloud-run)
23. [hflps://github.com/GoogleCloudPlatform/agent-stafier-pack](https://github.com/GoogleCloudPlatform/agent-stafier-pack)
24. Сократис Кафиакис, 2024, «GenAI в производстве: MLOps или GenAIOps?». Доступно по адресу: [hflps://medium.com/google-cloud/genai-in-production-mlops-or-genaiops-25691c9becd0](https://medium.com/google-cloud/genai-in-production-mlops-or-genaiops-25691c9becd0).
25. Гуанъя Лю, Суджай Соломон, март 2025 г. «Наблюдаемость агентов ИИ — развивающиеся стандарты и передовой опыт». Доступно по адресу: [hflps://opentelemetry.io/blog/2025/ai-agent-observability/](https://opentelemetry.io/blog/2025/ai-agent-observability/).
26. [hflps://discuss.google.dev/t/agents-are-not-tools/192812](https://discuss.google.dev/t/agents-are-not-tools/192812)
27. Дамьян Массон и др., 2024 г., «DirectGPT: интерфейс прямого управления для взаимодействия с большими языковыми моделями». Доступно по адресу: [hflps://arxiv.org/abs/2310.03691](https://arxiv.org/abs/2310.03691).
28. MCP UI — это система управления пользовательским интерфейсом с помощью инструментов MCP [hflps://mcpui.dev/](https://mcpui.dev/).
29. AG UI — это протокол управления пользовательским интерфейсом посредством передачи событий и, дополнительно, общего состояния [hflps://ag-ui.com/](https://ag-ui.com/).
30. A2UI — это протокол генерации пользовательского интерфейса посредством структурированного вывода и передачи сообщений A2A [hflps://github.com/google/A2UI](https://github.com/google/A2UI).
31. [hflps://cloud.google.com/vefiex-ai/generative-ai/docs/models/gemini/2-5-flash-live-api](https://cloud.google.com/vefiex-ai/generative-ai/docs/models/gemini/2-5-flash-live-api).
32. [hflps://saif.google/focus-on-agents](https://saif.google/focus-on-agents).
33. [hflps://simonwillison.net/series/prompt-injection/](https://simonwillison.net/series/prompt-injection/).
34. [hflps://storage.googleapis.com/gweb-research2023-media/pubtools/1018686.pdf](https://storage.googleapis.com/gweb-research2023-media/pubtools/1018686.pdf).
35. [hflps://spiffe.io/](https://spiffe.io/).
36. [hflps://openreview.net/pdf?id=l9rATNBB8Y](https://openreview.net/pdf?id=l9rATNBB8Y).
37. [hflps://google.github.io/adk-docs/safety/](https://google.github.io/adk-docs/safety/).

38. [hflps://google.github.io/adk-docs/callbacks/design-patterns-and-best-practices/#guardrails-policy-enforcement](https://google.github.io/adk-docs/callbacks/design-patterns-and-best-practices/#guardrails-policy-enforcement)
39. ТКТК
40. [hflps://cloud.google.com/security-command-center/docs/model-armor-overview](https://cloud.google.com/security-command-center/docs/model-armor-overview)
41. [hflps://cloud.google.com/vertex-ai/generative-ai/docs/provisioned-throughput/overview](https://cloud.google.com/vertex-ai/generative-ai/docs/provisioned-throughput/overview)
42. [hflps://cloud.google.com/run/sla](https://cloud.google.com/run/sla)
43. [hflps://github.com/CharlesQ9/Self-Evolving-Agents](https://github.com/CharlesQ9/Self-Evolving-Agents)
44. Juraj Gottweis и др., 2025, «Ускорение научных прорывов с помощью со-ученого с искусственным интеллектом». Доступно по адресу: [hflps://research.google/blog/accelerating-scientific-breakthroughs-with-an-ai-co-scientist/](https://research.google/blog/accelerating-scientific-breakthroughs-with-an-ai-co-scientist/).
45. Deepak Nathani и др., 2025, «MLGym: новая структура и эталон для продвижения исследовательских агентов ИИ», доступно по адресу: [hflps://arxiv.org/abs/2502.14499](https://arxiv.org/abs/2502.14499).